DOCTOR OF PHILOSOPHY

Effective monitoring of slow suspicious activities on computer networks

Kalutarage, Harsha

Award date:
2013

Awarding institution:
Coventry University

Effective monitoring of slow suspicious activities on computer networks

Harsha Kumara Kalutarage

Digital Security & Forensics (SaFe) Research Group
Department of Computing, Faculty of Engineering & Computing
Coventry University, UK

A thesis submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy in Cyber Security

Date: November 15, 2013

Abstract

Slow and suspicious malicious activities on modern computer networks are increasingly hard to detect. An attacker may take days, weeks or months to complete an attack life cycle. A particular challenge is to monitor for stealthy attempts deliberately designed to stay beneath detection thresholds. This doctoral research presents a theoretical framework for effective monitoring of such activities. The main contribution of this work is a scalable monitoring scheme proposed in a Bayesian framework, which allows for detection of multiple attackers by setting a threshold using the Grubbs' test. Second contribution is a tracing algorithm for such attacks. Network paths from a victim to its immediate visible hops are mapped and profiled in a Bayesian framework and the highest scored path is prioritised for monitoring. Third contribution explores an approach to minimise data collection by employing traffic sampling. The traffic is sampled using the stratification sampling technique with optimum allocation method. Using a 10% sampling rate was sufficient to detect simulated attackers, and some network parameters affected on sampling error. Final contribution is a target-centric monitoring scheme to detect nodes under attack. Target-centric approach is quicker to detect stealthy attacks and has potential to detect collusion as it completely independent from source information. Experiments are carried out in a simulated environment using the network simulator $NS3$. Anomalous traffic is generated along with normal traffic within and between networks using a Poisson arrival model. Our work addresses a key problem of network security monitoring: a scalable monitoring scheme for slow and suspicious activities. State size, in terms of a node score, is a small multiple of number of nodes in the network and hence storage is feasible for very large scale networks.

Effective monitoring of slow suspicious activities on computer networks

Harsha Kumara Kalutarage

Digital Security & Forensics (SaFe) Research Group

Department of Computing, Faculty of Engineering & Computing

Coventry University, UK

A thesis submitted in partial fulfilment of the requirements for the Degree of Doctor of Philosophy in Cyber Security

Date: November 15, 2013

I would like to dedicate this thesis to my loving parents and my family...

Acknowledgements

No programme of PhD work is ever done alone, so I must acknowledge the help and assistance of many people.

I would like to express my deepest gratitude to my Director of Studies Dr. Siraj Shaikh, who has the attitude and the substance of a genius, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. Without his guidance and persistent help this dissertation would not have been possible.

I would like to thank my supervisors Dr.Qin Zhou and Professor Anne James for guiding me at various points in this PhD journey and providing an excellent feedback on the final draft of this dissertation. A big thank goes to Coventry University for offering me a scholorship to support this study. Special thanks go to Prof.Paddy Krishnan, Bond University, Australia and Dr.Indika Wickramasinghe, Eastern New Mexico University, US for spending their valuble time for providing me feedbacks on mathematical basis of this work. In addition, a big thank to all PRP members who gave me constructive feedbacks at each PRP meetings. I must thank to all anonymious reviewers of our paper submissions produced from this work.

I would also like to thank my parents, brothers, and friends in my home country and Coventry University. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to thank my daughter Suvithma and wife Geethi for their patient and love during last few years. They were always there cheering me up and stood by me through the good and bad times.

Abstract

Slow and suspicious malicious activities on modern computer networks are increasingly hard to detect. An attacker may take days, weeks or months to complete an attack life cycle. A particular challenge is to monitor for stealthy attempts deliberately designed to stay beneath detection thresholds. This doctoral research presents a theoretical framework for effective monitoring of such activities. The main contribution of this work is a scalable monitoring scheme proposed in a Bayesian framework, which allows for detection of multiple attackers by setting a threshold using the Grubbs' test. Second contribution is a tracing algorithm for such attacks. Network paths from a victim to its immediate visible hops are mapped and profiled in a Bayesian framework and the highest scored path is prioritised for monitoring. Third contribution explores an approach to minimise data collection by employing traffic sampling. The traffic is sampled using the stratification sampling technique with optimum allocation method. Using a 10% sampling rate was sufficient to detect simulated attackers, and some network parameters affected on sampling error. Final contribution is a target-centric monitoring scheme to detect nodes under attack. Target-centric approach is quicker to detect stealthy attacks and has potential to detect collusion as it completely independent from source information. Experiments are carried out in a simulated environment using the network simulator $NS3$. Anomalous traffic is generated along with normal traffic within and between networks using a Poisson arrival model. Our work addresses a key problem of network security monitoring: a scalable monitoring scheme for slow and suspicious activities. State size, in terms of a node score, is a small multiple of number of nodes in the network and hence storage is feasible for very large scale networks.

Publications

Following is list of research publications resulting from this doctoral effort.

1. Harsha K. Kalutarage, Siraj A. Shaikh, Qin Zhou, and Anne E.James. Monitoring for Slow Suspicious Activities using a Target centric Approach. In A. Bagchi and I. Ray editors, ICISS 2013, Kolkata, India, December 2013. Proceedings, volume 8303 of Lecture Notes in Computer Science, pages 163-168. Springer, 2013
2. Harsha K. Kalutarage, Siraj A. Shaikh, Qin Zhou, and Anne E.James. Tracing sources of anonymous slow suspicious activities. In Lopez, Javier and Huang, Xinyi and Sandhu, Ravi, editors, Network and System Security, NSS2013, Madrid, Spain, June 2013. Proceedings, volume 7873 of Lecture Notes in Computer Science, pages 122-134. Springer, 2013
3. Kalutarage, H.K., Shaikh, S.A., Zhou, Q., James, A.E.: How do we effectively monitor for slow suspicious activities? In Maritta Heisel and Eda Marchetti editors, International Symposium on Engineering Secure Software and Systems (ESSoS-DS 2013), Inria, Paris (Rocquencourt), France, February/March, 2013, CEUR Workshop Proceedings, pages 36-40. CEUR-WS.org, 2013.
4. Kalutarage, H.K.; Shaikh, S.A.; Qin Zhou; James, A.E., Sensing for suspicion at scale: A Bayesian approach for Cyber conflict attribution and reasoning, 4th International Conference on Cyber Conflict (CYCON 2012), NATO Cooperative Cyber Defence Centre of Excellence, Tallinn, Estonia, June 2012, pages 1-19, In IEEE CYCON 2012 Proceedings.

Contents

List of Figures

Chapter 1

Introduction

It has been estimated that the number of intrusion attempts over the entire Internet was in the order of 25 billion per day in year 2003 (Yegneswaran *et al.*, 2003) and continues to increase (McHugh, 2001). While methods and technologies for securing networks against intruders continue to evolve, system penetration attempts continue to occur and go unnoticed until it is too late. This is due to a variety of reasons including attackers continue to find novel and more sophisticated techniques to breach the systems and the size and complexity of networks is ever increasing (Patcha & Park, 2007; Yegneswaran *et al.*, 2003).

Launching *slow rates attack* is one of such sophisticated techniques used by skilful attackers to avoid detection. This allows for malicious activity to blend into the network as noise to never exceed detection thresholds and to exhaust the detection system state. The focus of this thesis is monitoring for slow attacks that occur over longer periods of time and which are difficult to detect using conventional detection methods largely developed for rapid attacks.

1.1 Motivation and main goals

Slow, suspicious and increasingly sophisticated malicious activities on modern networks are incredibly hard to detect. An attacker may take days, weeks or months to complete the attack life cycle against the target host. A particular challenge is to monitor for such attempts deliberately designed to stay beneath

detection thresholds. Attacker tactics such as source collusion and source address spoofing are common, and tools and techniques to launch such attacks are widely available (Hackers, 2013; Mullins, 2013). The aim of our study is to establish a theoretical framework for an effective monitoring scheme for slow suspicious activities. The research objectives of the study can be broken down to following:

1. How to effectively detect the sources of slow suspicious activities?

   The main goal is to establish a scalable method for detecting sources of such activities. A review of different possible methods and particularly an investigation of whether a Bayesian approach is feasible to achieve this is performed.

2. Can sampling techniques be employed to provide a lightweight monitoring scheme?

   Traffic volumes will continue to increase and huge volumes of traffic may consume more resources of the monitoring infrastructures. This makes it ever more difficult to provide lightweight schemes for near-real time monitoring. We investigate the feasibility of reducing the traffic volume needed to be analysed by the proposed algorithm without degrading the quality of detections.

3. How to effectively detect the target of slow suspicious activities?

   The use of botnets and distributed attack sources make it very difficult to attribute attacks. We investigate methods to profile such targets instead of sources of activities to overcome source collusion.

1.2 Scope of the study

The main purpose of this work is to develop a conceptual framework for a possible solution for this critical problem. Our approach should be considered as complementary to current intrusion detection deployments. It will serve as an early warning system for slow suspicious activities over networks that warrant further investigations.

1.2.1 Terminology

Slow activity - if any attack activity is not alarmed by existing methods only due to stretching its steps over longer times, then it is identified as a slow activity for this study.
Node - anything in terms of identities which can be a user, machine, account number or a location (physical or virtual), essentially a source or a destination point of a potential attack.
Suspicious event - any type of user activities that can be triggered by existing IDSs techniques are considered as suspicious events for this study. Sending a connection request to a node which is not in the network and entering incorrect user-name/password are just two examples for such an event.
Scalability - ability of a computer system to continue to function well when its context is changed in size or volume.

1.3 Thesis outline

The rest of the thesis is organised as follows. Chapter 2 provides an overview of intrusion detection in computer systems, and explains why typical methods for intrusion detections cannot be employed in slow activity monitoring. It reviews the incremental anomaly detection and visualisation techniques which is used as a data reduction method for anomaly detection. An introduction to IDSs evaluation datasets and techniques are also provided.

Chapter 3 proposes a novel algorithm for slow activity monitoring. The main problem is broken down into two sub problems, and each sub problem is addressed separately. This chapter identifies Bayesian approach as a method for information fusion to address the problem, and examines the effectiveness of such an approach under different network conditions: multiple attackers, traffic volume and subnet configuration. Theoretical account of the approach and detailed discussions of experimental results are provided. The main contribution of this chapter is a novel algorithm for slow activity monitoring.

Chapter 4 discusses the attribution problem, and provides an anomaly based adaptive method for tracing down the sources of slow suspicious activities in com-

puter networks. A theoretical account of the approach and experimental results are provided. The main contribution of this chapter is the tracing algorithm.

Chapter 5 examines the feasibility of employing traffic sampling with our monitoring algorithm as a method of data reduction without degrading the quality of detection. A study of how the design of the network affects on sampling error is also presented.

Chapter 6 presents a target-centric monitoring scheme which utilises destination information of activities disregarding its logic source. The purpose of this chapter is to convey the core principle.

Finally, Chapter 7 summarises the main contributions of this thesis, presents concluding remarks and ideas for further research.

Chapter 2

Background

This chapter explores the literature on network security monitoring, and explains why typical monitoring methods cannot be employed in slow activity monitoring.

2.1 Intrusion detection

Computer security threats have been comprehensively studied since the seminal report written by Anderson (1980). *Host based* and *network based* are a typical classification of IDSs based on the monitoring location. *Signature (misuse) detection* and *anomaly detection* are the two main common classifications of intrusion detections based on the detection technique employed.

A host based IDS will monitor resources such as system logs, file systems and disk resources; whereas a network based system monitors the data passing through a network. Host based systems are incapable of detecting distributed and coordinated attacks. Network based systems aim at protecting the entire networks against intrusions by monitoring the network traffic either on designed hosts or specific sensors and thus can protect simultaneously a large number of computers running different operating systems against remote attacks such as distributed denial of service attacks, propagation of computer worms (Tavallaee *et al.*, 2008).

Signature detection systems try to find attack signatures in the monitored resource. Encoding the knowledge about patterns in the data flow (i.e. in form

of specific signatures) is the aim of signature detection. Most attacks leave a set of signatures in audit trails (or in network packets) is a common belief of this approach. Signature based IDSs exploit signatures of known attacks. Hence attacks are detectable only if these signatures can be identified by analysing the audit trails (Tavallaee *et al.*, 2008). Such systems require frequent updates of signatures for known attacks. Detecting known attacks with lower false alarm rates is a significant advantage of this approach. However, if signatures are not available or unknown (with newer or unknown attacks) then this approach is failed.

In contrast, addressing the weaknesses of signature detection (Tavallaee *et al.*, 2008), the concept of anomaly detection was formalized in the seminal report written by Denning (1987). Anomaly detection systems typically rely on knowledge of normal behaviour. When the actual system behaviour deviates from the normal profiles in the system an anomaly is flagged. This anomaly could be because of either innocent or threat event in the system (see sections 2.1.1 and 3.3.1.1). Though existing anomaly based IDSs can be employed in finding and preventing known as well as unknown (zero day) attacks, they have many shortcomings such as high rate of false alarms, and are failure to scale up to gigabyte speeds (Patcha & Park, 2007). A structured and comprehensive survey on network security tools and systems which are useful for security researchers can be found in (Hoque *et al.*, 2013). Hoque *et al.* classify existing tools, including information gathering and launching tools, and provide an analysis of their capabilities. A comparison among different Network IDSs is also provided in the same report.

2.1.1 Evidential scenario

Computer systems are dynamic systems having many components such as servers, clients, routers, switches, firewalls and IDSs. At each time interval, these components produce large amount of event based data which, in principle, can be collected and used for security analysis. The signature elements of an attack is scattered spatially and temporally, and often embedded within the totality of events of the distributed systems. This evidence distribution can be depicted as in Figure 2.1, and has following characteristics.

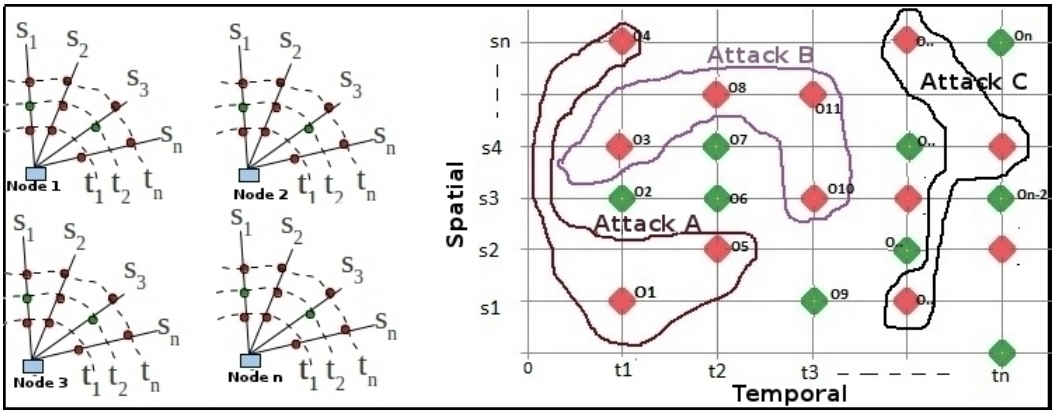

Figure 2.1: Evidential scenario. s_i - information source i, t_i - time point i, o_i - observation i, n_i - node i.

- Two dimensional event distribution: *temporal* and *spatial*.
- Four possible types of events:

  Malicious - events occurred certainly due to an intrusion attempt (represented by red diamonds in Figure 2.1).

  Innocent - events occurred certainly due to a legitimate activity (represented by green diamonds in Figure 2.1). Hence monitoring systems will not report them.

  Suspicious - events occurred due to either a malicious attempt or a legitimate activity. They can be considered as either red or green diamonds based on the motivation behind the activity. Events such as an execution of cmd.exe, a multiple loging failure attempts, an overwhelming number of ICMP unreachable messages are examples for such events.

  Not reported - events occurred as a part of malicious attempt, but not reported by the monitoring system due to its incapability to detect and alert them.

- Source anonymity: The exact *origin* of the event (who did that?) is not always certain.

As mentioned above, every suspicious event alerted by a monitoring system cannot be treated certainly as a part of malicious attempt. For example, a major router failure could generate many ICMP unreachable messages while some viruses and worms (e.g. CodeRed and Nimda) generate the same in probing process. By just looking at such an event you cannot simply judge its motivation that it is a part of malicious attempt or not. This uncertainty which we called *Motivation* needs to be acknowledged. Our approach in Chapter 3 for slow attack monitoring is capable of acknowledging this.

There is no guarantee on publicly visible source of an event is to be the true source. For example, employing various proxy methods and zombie nodes (e.g. bots), manipulation of TCP/IP elements (e.g. IP Spoofing), using random routing or even proliferation of weakly encrypted wireless networks let an attacker to be anonymous. Hence source oriented monitoring schemes are often vulnerable to this *Source* uncertainty. Chapter 6 proposes an alternative monitoring approach to overcome this.

2.1.2 Signature elements

In order to identify a trace of an attack, an efficient correlation algorithm is an essential. For example, let $\{S_1, S_2, S_3, ..., S_n\}$ is a sequence of signature elements which characterises a multi step attack M. In order to detect M within a given dataset (a trace), the detector should be able to observe the same sequence (or an acceptable level of similar pattern of S_is) within the dataset. That recognition is not easy as it looks like as S_is will not come into the scene as a consecutive sequence, one after other according to the same manner defined in the signature definition database.

Signature elements are randomly distributed, spatially and temporally, among other types of events mentioned above. Like other pattern recognition problems, an attack scenario signature (a pattern of signature elements which uniquely describes the attack) is needed to distinguish a given attack (say A) from other attacks (B and C) and from normal network activities (see Figure 2.1). With large amount of event data, the critical challenge is how to correlate these events (O_is) across observation space and time to detect and track various attack sce-

narios such as A, B and C. The detection accuracy relies on the accuracy of scenario signature as well as the accuracy of event correlation (Jiang & Cybenko, 2004). Defining scenario signature is possible for known attacks, but for unknown attacks it is impossible.

2.1.3 Holding event data

State full security devices such as firewalls and routers are capable of maintaining the history of events for security purposes, but limited to a very short history, usually for few minutes back from the present state (Sheth & Thakker, 2011; Wool, 2006). After that specific time period the information is discarded for using the limited storage space for capturing most recent events. Hence holding event data for very long times (i.e. for larger observation windows) is impossible due to the current computational constraints.

On the other hand it is unknown in advance that how long a particular suspicious event to be retained to correlate with proceeding events occurred. For example, in order to detect attack M above, it is needed to hold all the event data until it recognises the complete sequence of signature elements $s_1, s_2, s_3, ..., s_n$. If the time gap between emergence of s_1 and s_n (*event horizon*) was few days, weeks or even months the problem getting only into its worst case, especially, in a high volume high speed environments. For example, capturing complete traffic with just a 512 kbps connection which is operating at a 50% of average utilization rate would end up with a 2.7GB in a day. Obviously it is impossible to store complete data of a modern enterprise network for analysis over a long time period.

As a solution, often monitoring systems simply log the history of connection information neglecting other traffic data such as the packet payloads. Although this reduces significantly the amount of data recorded, it will still result in thousands of log entries per day which is an unmanageable amount for a network professional to read line by line (van Riel & Irwin, 2006b). As described in the Cisco's Security Information Event Management (SIEM) deployment guide (CSIEM, 2013), managing the sheer volume of raw logs and events gathered from various devices and applications throughout the enterprise can be a very costly effort in terms of time, bandwidth and computational resources,

and therefore organizations need a unified view of the state of network security in a single dashboard.

2.2 Slow attacks

It is essential to exploit more evidence (signature elements) from large number of network events to get better detection accuracy for some attacks while other attacks can be detected using a single packet or a connection (Jiang & Cybenko, 2004; Lazarevic *et al.*, 2003). Often, network based intrusions signatures are state-full and usually require several pieces of data to match an attack signature. If the length of the *event horizon* (time amount from the initial data piece to the final data piece needed to complete the attack signature) is longer, IDSs cannot maintain state information indefinitely without eventually running out of resources. This helps slow attackers to hide behind noise and other traffic. Most current approaches do not track activity over an extended period of time due to computational constraints and disk storage requirements. In order to avoid detection, an attacker can stretch her attack attempts (distribute signature elements) across temporal or/and spatial spaces.

An attack which stretches deliberately its steps (events) over temporal space is defined as a *slow attack* for this work while an attack which stretches its steps over spatial space is defined as a *distributed attack*. If an attack distributed its signature elements over both dimensions then it is known as a *slow distributed attack*. To best of our knowledge, there is no clear boundary to distinguish slow attacks from typical rapid attacks based on the temporal aspects of attack life cycle. Any suspicious attempt which is deliberately operated slowly to stay beneath the detection threshold is identified as a *slow event* for this study. A combination of such events that is sufficient to complete an attack life cycle form a slow attack. A node used to launch such attack is known as a *slow attacker* in this work.

As mentioned above for detection of typical attack types the IDSs consider the event stream as a function of time and use signature or anomaly based methods against the event flow. This is usually not an issue when the progression of attack events is rapid. Most current IDSs are capable of doing this with signifi-

cant performances as traditional attackers relied more on rapid attacks. However increasingly nowadays attackers are trying to remain undetected and to steal information over and over again, adopting a much more patient type of structure to compromise a network. An attacker may take days, weeks or months to complete the attack life cycle against the target host. Attacks may blend into the network noise in order to never exceed detection thresholds and to exhaust detection system state. Therefore, as computer networks scale up in terms of number of nodes and volume of traffic analysing slow attack activities, deliberately designed to stay beneath thresholds, becomes ever more difficult using typical methods designed for rapid attack detections. Appendix A includes a real world simple attack scenario which can hide beneath thresholds by means of slow activity rates. When a slow attack is progressing, scarcity of attack data within a short period of time allows an attacker to avoid detection. For example Hackers (2013); Mullins (2013) present tools and technique to perform such attacks. Advanced Persistent Threats (APTs) is an example threat model for a slow attack. This thesis focuses on slow attacks only. All other types of attacks are beyond the scope of this study.

2.2.1 Advanced persistent threat

Advanced Persistent Threats (APTs) require a high degree of stealthiness over a prolonged duration of operation in order to be successful. Therefore compromised systems continue to be of service even after key systems have been breached and initial goals reached (DAMBALLA, 2013).

Advanced - Criminal operators behind the threat utilize the full spectrum of computer intrusion technologies and techniques. They combine multiple attack methodologies and tools in order to reach and compromise their target.

Persistent - Criminal operators give priority to a specific task, rather than opportunistically seeking immediate financial gain. The attack is conducted through continuous monitoring and interaction in order to achieve the defined objectives. A "low-and-slow" approach is usually more successful.

Threat - means that there is a level of coordinated human involvement in the attack, rather than a mindless and automated piece of code. The criminal op-

erators have a specific objective and are skilled, motivated, organized and well funded.

2.3 Anomaly detection

Intrusions detection can be performed based on information collected from computer systems. An intrusion is different from the normal behaviour of the system, and hence anomaly detection techniques are applicable in intrusion detection domain (Chandola *et al.*, 2009). This has been formulated in (Giacinto & Roli, 2002) as a pattern recognition problem. Finding nonconforming patterns or behaviours in data is referred as anomaly detection. Anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities or contaminants are various terms used in different application domains to denote these patterns. Intrusive activity is always a subset of anomalous activity is the ordinary belief of this idea (Delooze & Kalita, 2006; Patcha & Park, 2007). The key challenge for anomaly detection in this domain is the huge volume of data, typically comes in a streaming fashion, thereby requiring on-line analysis. The anomaly detection techniques need to be computationally efficient to handle these large sized inputs. When there is an intruder who has no idea of the legitimate user's activity patterns, the probability that the intruder's activity is detected as anomalous is high. There are four possibilities with none zero probabilities that can be happened in such a situation (Bhuyan *et al.*, 2011):

- Intrusive but not anomalous: The detection system may fail to detect this type of activity since the activity is not anomalous. This is known as a false negative because it falsely reports the absence of an intrusion when there is one.

- Not intrusive but anomalous: The detection system reports an activity as intrusive, when it is not, since it produces an anomaly. This is known as a false positive because an intrusion detection system falsely reports intrusions.

- Not intrusive and not anomalous: These are true negatives; the activity is not

intrusive and should not be reported as intrusive.

- Intrusive and anomalous: These are true positives; the activity is intrusive and should be reported as such.

Good detection systems should always try to reduce the probability of occurring false positives and false negatives while trying to increase the probability of true negative and true positives. Since the data amounts to millions of data objects, a few percent of false alarms can make analysis overwhelming for an analyst. But it is very challenging in network security monitoring to achieve lesser false alarms, due to the nature of anomalies keeps changing over time as the intruders adapt their network attacks to evade the existing intrusion detection solutions.

Clustering Based, Information Theoretic, Neural Networks, Nearest Neighbour based, Rule-based Systems, Spectral Analysis, Statistical (parametric and non-parametric) are examples of anomaly detection techniques used for network intrusion detections (Chandola *et al.*, 2009). Since late 90's, a significant number of anomaly based intrusion detection approaches have been proposed, but lot of them are general in nature and quite simple (Patcha & Park, 2007; Spafford & Kumar, 1994; Chandola *et al.*, 2009). Only a few of them are incremental approaches which could be useful in evidence accumulation, but are not scalable. An incremental approach updates profiles dynamically incorporating new profiles as it encounters them (Bhuyan *et al.*, 2011). Bhuyan *et al.* (2011) is an exhaustive survey of incremental anomaly detection approaches which concludes that most current approaches have high rate of false alarms, are non-scalable, and are not suitable for deployment in high-speed networks.

Supervised, semi-supervised and unsupervised are three possible types of anomaly detection methods. This classification is based on availability of data labels. Techniques that operate in supervised mode assume that availability of a training data set which has labelled instances for normal as well as anomaly classes. Semi-supervised mode depends on a training data which has labelled instances only for the normal class while unsupervised mode not requiring any training data. Published works for each of these three types are available. Getting a labelled set of anomalous data instances that covers all possible type of anomalous behaviour is more difficult in this domain than getting labels for nor-

mal behaviour (Chandola *et al.*, 2009). Both normal and anomalous behaviour are often dynamic (e.g. new types of anomalies might arise for which there is no labelled training data), and hence unsupervised mode (or semi-supervised mode running in unsupervised mode) is encouraged and widely acceptable in this domain (Chandola *et al.*, 2009). Our proposed anomaly detection approach in this thesis is unsupervised. Therefore discussions on supervised and semi-supervised approaches are excluded from this section, and interesting readers are invited to refer (Chandola *et al.*, 2009).

As mentioned above our work is an unsupervised incremental approach which does not require training data. Most existing unsupervised anomaly detection approaches are clustering based (Bhuyan *et al.*, 2011; Hsu & Huang, 2008; Zhong & Li, 2008a), a technique uses to group similar objects, and packet based clustering is used as the means of detection. Clustering plays a vital role in analysing data in detection of anomalies by identifying various groups as either belonging to normal or to anomalous categories. Most commonly used clustering techniques are: partitioning-based (Ren *et al.*, 2008), hierarchical (Zhong & Li, 2008b), density-based (Burbeck & Nadjm-Tehrani, 2007), and grid-based techniques. The proposed approach in this thesis does not use clustering as the means of detection. Clustering is quite difficult when dealing with mixed data observed from different types of observation spaces (multivariate) (Hsu & Huang, 2008; Zhong & Li, 2008a). Usually clustering methods partition the dataset into two or more clusters and then label each clusters as normal or anomalous. Obtaining accurate and representative labels are challenging in this domain (Bhuyan *et al.*, 2011). Further, as described in (Chandola *et al.*, 2009) though testing phase is usually fast in clustering based methods, the computational complexity for clustering the data often forms bottlenecks, especially when $O(N^2d)$ clustering algorithms are used. Many techniques detect anomalies as a by-product of clustering, and hence are not optimized for anomaly detection. Therefore existing clustering based approaches are not scalable and not fitted with slow activity detection, and hence excluded from this discussion. A comprehensive overview of the research on anomaly detections can be found in (Chandola *et al.*, 2009). Bhuyan *et al.* (2011) is a comprehensive survey on incremental anomaly detection techniques which concludes that incremental approaches reduce memory utilization,

are faster, exhibit higher detection rate, and improve real time performance.

2.3.1 Incremental approaches

Chivers *et al.* (2009, 2013) provide a scalable solution for insider detection in a Bayesian framework by maintaining incremental node scores for each node in the network. All nodes, in fact origin of activities, in the network are profiled by the time and distinguished between anomaly and normal behaviours by setting a control (baseline). If the cumulative node score (i.e. node scores accumulated over the time) of a particular node is deviated from the predefined control, an anomaly is declared and that node is identified as a slow suspicious insider. The major drawback of this approach is setting a predefined control as the baseline. Setting predefined controls is very challenging in network security monitoring. As described in section 3.3.3.1, in a network scenario, normal behaviour keeps evolving and a current notion of normal behaviour might not be sufficiently representative in the future.

Kandias *et al.* (2010) combine tools and techniques of computer science and psychology for insider threat prediction. Real time user's technological traits (i.e. usage information) are integrated with data obtained from psychometric tests (i.e. user characteristics) to profile users, and hence to analyse their misbehaviours. Kandias *et al.*'s model combines the above information, categorizes users, and identifies those that require additional monitoring as they can potentially be dangerous for the organization as insiders.

Greitzer *et al.* (2009) is a similar work to (Kandias *et al.*, 2010). It provides a research framework for testing hypothesises for insider threats by integrating employee data with traditional cyber security audit data. Their approach is based on pattern recognition and model-based reasoning. Reasoner is the pattern recognition component which analyses the large amount of noisy data to distinguish variations from norms. Data is processed using a dynamic Bayesian network which calculates belief levels assigned to indicators and assessed the current indicators with the combination of previously assessed indicators to determine the likelihood of behaviours that represent threats. Probabilities are assigned for the Reasoner through expert knowledge. Authors have chosen simulation method to

evaluate the proposed approach realising the difficulty to find real cases in this domain. However this study has limited its focus on predicting individual employee's (human) behaviour rather than node behaviours in computer networks.

The interesting idea proposed in (Kandias *et al.*, 2010; Greitzer *et al.*, 2009) is to incorporate wider range of information into the monitoring which we believe very important. This idea increasingly becomes popular among security community (Davidoff & Ham, 2012). Kandias *et al.*; Greitzer *et al.* propose to combine some features of technical solutions (e.g. system call alerts, intrusion detection system alerts, honey pot, systems logs, etc) with data drawn from psychometric tests (e.g. predisposition, stress level, etc). However, including psychometric tests data is possible in principle, but in practice determining sources of events and their degrees of potential (i.e. characteristics) is the real challenge. Determining the source of event is not straightforward as attackers usually use various methods for getting anonymous. We discuss the anonymity problem, a critical research area in this domain in Chapter 4. When addressing non human threats these approaches may face difficulties due to the psychological profiling components. Hence they are highly organisational dependent, and expertise knowledge is needed to fine-tune the model in order to fit with new environments. Importantly those approaches are not scalable as when the operating environment is changed, the model needs to rebuild.

Eberle *et al.* (2010) is a graph-based anomaly detection (GBAD) systems which discovers anomalous instances of structural patterns in data that represent entities, relationships and actions. GBAD is applied to datasets that represent the flow of information between entities, as well as the actions that take place on the information. Authors claim GBAD can be applied to tackle several security concerns including identifying violation of system security policies and differentiating suspected nasty behaviour from normal behaviour. The graph substructures are analysed for discovering three types of graph-based anomalies: modifications, insertions and deletions. The Minimum description length heuristic is used to discover the best substructure of the given graph and, for anomalous modifications, cost of transformation and frequency values for all of the instances of that substructure are calculated using an inexact matching algorithm. These values are used to distinguish the anomalies pattern from the normative patterns. For

anomalous insertions, all extensions (instead of all instances) to the normative pattern are examined probabilistically to determine if there is an instance that extended beyond its normative structure. For anomalous graph deletions, all of the instances of ancestral substructures are examined and transformation cost and frequency are calculated. Hence anomaly patterns are distinguished from the normative patterns. Proposed approach has tested against Enron e-mail dataset, a sample of cell-phone traffic and simulated order-processing model using OMNeT++. Authors have acknowledged the need of improvement to the GBAD system by reducing the time spent for main computational bottleneck. Hence proposed approach is not suitable for real time monitoring, specially for high volumes and high speeds environments. Obviously when the data set is huge graph based approaches face difficulties.

As mentioned in (Tavallaee *et al.*, 2008), though machine learning techniques can be employed on detecting network anomalies, they are faced with problems. Leaning algorithms are failed in this domain due to the behavioural non-similarity in training and testing data. But, signal processing techniques can be successfully applied. Hence, a novel covariance matrix based approach is proposed in (Tavallaee *et al.*, 2008) for detecting network anomalies using the correlation between groups of network traffic samples. The signs in the covariance matrix of a group of sequential samples is compared against the training (normal) data, and the number of differences is counted to compare it with a specified threshold. The major drawback of this approach, as of many supervised approaches, is depending on predefined normal behaviours through training datasets. Defining normality in advanced is a challenge as characteristics of normality is not stationary. Also, newly generated threats will not be characterized by drawn training datasets and hence these threats will not be detected. Frequently training is possible, but how to determine the need of training without knowing new threat is existed. Other interesting incremental supervised (and semi supervised) anomaly detection approaches such as (Yu & Lee, 2009; Laskov *et al.*, 2006; Ren *et al.*, 2008; Khreich *et al.*, 2011; Lu *et al.*, 2011; Yi *et al.*, 2011; Rasoulifard *et al.*, 2008; Burbeck & Nadjm-Tehrani, 2007) will not be discussed here. In general, there are two major issues associated with supervised anomaly detections (Bhuyan *et al.*, 2011): 1-numbers of anomalous instances are far fewer than the number of nor-

mal instances in the training dataset and 2-obtaining accurate and representative labels are challenging. These issues are discussed in detailed in (Joshi *et al.*, 2001; Theiler & Cai, 2003).

2.3.2 Visualisation

Managing overwhelming amount of data is a major challenge in network security monitoring. Network monitoring becomes unmanageable (bulky) when working with a huge quantity of log entries and intrusion alerts (van Riel & Irwin, 2006a). Having to go through huge amount of text data (packet traces, log files, etc) to gain insight into networks is a common, but a tedious and an untimely, task as terabytes of information in each day is usual in a moderate sized network (Ball *et al.*, 2004). Works summerised in this section have been used information visualisation as a data reduction method for anomaly detection.

Since most of current network monitoring tools do not provide rapid overview of the security state of networks, especially during a crisis as most of them are textbased, Ball *et al.* (2004) present a novel computer security visualisation tool which visualises a quick overview of current and recent communication patterns in the network. Authors propose to visualise packet flows in the network assuming it will help network professionals to have an accurate mental model of what is normal on their own network and hence to recognise abnormal traffic.

Fisk *et al.* (2003) claim that "the human perceptual and cognitive system comprises an incredibly flexible pattern recognition system which can recognise existing patterns and discover new patterns, and hence recognizing novel patterns in their environment which may either represent threats or opportunities". Hence, authors proposed a 3-D immersive network monitoring system underlining that proposed system allows users to survey day's worth of traffic in minutes. They argue that using memorable visual images to represent traffic makes easy for non-skilled people to observe events that only skilled analysts were sensitive before.

van Riel & Irwin (2006a) use graphical depictions to cover up alert information over raw network traffic by combining dedicated sensor network monitoring (such as network telescopes or honeynets) with visualisation. All observed traffic is treated as mistrust as no genuine clients (services) reside in dedicated sensor

networks. Yin *et al.* (2004) provides traffic visualization design, similar to (Ball *et al.*, 2004; Fisk *et al.*, 2003; van Riel & Irwin, 2006a), to enhance administrator's ability on detecting anomalous traffic patterns.

van Riel & Irwin (2006b) claim that visual analysis facilitates the rapid review and correlation of events utilizing human intelligence in the identification of patterns. They acknowledge that data scalability has complicated to distinguish between suspicious and legitimate activities in computer networks, and proposes a novel investigating method for network monitoring combining network telescope traffic and visualisation, concluding that visualisation saliently suggest anomalous patterns. They criticise existing IDDs as signature based IDSs can only adopt uncovering known attacks while anomaly based (or hybrid) IDSs tend to have higher rates of false alarms. It is essential IDSs to be extremely accurate as most traffic is innocent in many situations. Crafting packets to generate a devastating number of alerts and disrupting traffic patterns are possible techniques that can be used by attackers deliberately to disturb intrusion detections by both signature and anomaly detections methods (van Riel & Irwin, 2006b).

The intended aim of network security measures is to reduce the effort expending by network professionals in uncovering suspicious network activity. But in practice this has became a challenge (bulky) when dealing with a huge amount of data such as large number of log entries and intrusion alerts. Note that in principle all above works acknowledge that visualisation (by means of graphs or animation) is useful in identifying anomalies patterns. But our understanding, though visualisation can be motivated on this as visual cognition is highly parallel and pre-attentive than the text or speech, it does little on slow attack monitoring. Just presenting raw data in graphical form would not be sufficient. For example visualising a traffic flow of a large network for a very long time will end up with a very complicated web of traffic flow. It would be very difficult to compare this with administrator's mental model of the netflow already made in mind. Therefore some kind of data reduction and simplification is needed before visualising security measures. As explained in Chapter 3, converting all information gathered through various information sources into a single node score (information fusion) is the simplification strategy we use in this work. In addition, standardization of security measures would help for better comparison as anomaly detection can be

highly sensitive to small changes of node behaviours.

2.4 Evaluation

Getting validity for a novel method is only possible through a proper evaluation. But in the domain of intrusion detections, evaluation of novel algorithms against real time network data is difficult (Bhuyan *et al.*, 2011; Kayacik *et al.*, 2005; KDD, 1999). Specially it becomes a real challenge if the focus is on slow activities. Current IDSs require expensive human input for either creating attack signatures or determining effective models for normal behaviour. Learning algorithms which derive models from training data for characterizing normal and attack behaviours provide an alternative to expensive human input (Kayacik *et al.*, 2005). But due to the extensive amount of data in networks and lack of availability of training datasets have limited the feasibility of using Learning algorithms in this domain.

2.4.1 Evaluation datasets

As it is very difficult to evaluate a novel algorithm based on live (or any raw) network traffic, very often simulation methods or some benchmark datasets are used by researchers for evaluations of their algorithms (Bhuyan *et al.*, 2011). In 1998, MIT's Lincoln Lab generated the DARPA training and testing datasets to evaluate IDSs (Lippmann *et al.*, 2000). Both datasets are included with attacks and background traffic. One year after, pre-processed DARPA training and testing data sets (known as the KDD99) is supplied (Burbeck & Nadjm-Tehrani, 2007). KDD99 is consisting of about four gigabytes of compressed binary tcpdump data from seven weeks of simulated network traffic. Simulated attacks fall into one of four categories: denial of service, remote-to-local (R2L), user-to-root(U2R), and surveillance (probing). Background traffic is simulated and the attacks are all known. The training set, consisting of seven weeks of labelled data, is available to the developers of IDSs. The testing set also consists of simulated background traffic and known attacks, including some attacks that are not presented in the training set. Kayacik *et al.* (2005) studies the contribution of 41 features (grouped into four categories as basic, content, time and connection

based features) in KDD99 datasets to attack detection. Features constructed from the data content of the connections are more important when detecting R2L and U2R attack types in KDD99 intrusion datasets (KDD, 1999), while time-based and connection based features are more important for detection of DoS (Denial of Service) and probing attack types (Lee & Stolfo, 1998).

LBNL is an evaluation dataset that can be obtained from Lawrence Berkeley National Laboratory (LBNL) in the USA. Traffic in this dataset is comprised of packet level incoming, outgoing, and internally routed traffic streams at the LBNL edge routers (Bhuyan *et al.*, 2011). The traffic is anonymous, and the attack rate is significantly lower than the background traffic rate. Endpoint Datasets, is a dataset that having the feature of traffic rates observed at the end points are much lower than those at the LBNL routers. To generate attack traffic, the analysts infected virtual machines on the endpoints with different malwares such as Zotob.G, Forbot-FU, Sdbot-AFR. For each malware, attack traffic of 15 minutes duration has been inserted in the background traffic at each end-point at a random manner (Bhuyan *et al.*, 2011; Spafford & Kumar, 1994). ISCX2012 is also a simulated data set for evaluating IDSs (Shiravi *et al.*, 2012). In this dataset, behaviours of certain group of users are abstracted into profiles, and then agents are programmed to execute them mimicking user activities. Attack scenarios are then simulated to express real world cases of malicious behaviours.

Network traces captured from live networks is also used for evaluation of IDS (Chandola *et al.*, 2009). The main advantage of this is that the results do not demonstrate any bias. However (Bhuyan *et al.*, 2011) claims, when network traces are available, they are often limited, and researchers evaluate their algorithms based on live network traffic cannot claim that their detection methods work in all situations. Also flow level data captured at the router level in online mode is called NetFlow data and quite useful for high-speed traffic analysis (Giacinto & Roli, 2002). Due to the loosing packet contents, some attack can be undetected when using only NetFlow data.

DARPA and KDD99 are the most widely employed data sets in intrusion detection domain. Despite the signicant contributions of DARPA and KDD99 datasets being the first standard corpora in this domain, their accuracy and ability to reflect real world conditions has been extensively criticized (Brown *et al.*,

2009; McHugh, 2000). Brugger & Chow (2007) states that detection performance of these data sets are low and false positive rates are unacceptable. Some scholars argue that these data sets are too old and not suitable for modern research approaches. They identify that simulated environment as the only convenience method at present in this domain. This data sensitivity has been restricted the research progression of this area.

Obviously, as mentioned above, existing datasets focus on evaluation of algorithms developed for typical quick attacks. They are not suitable for evaluating algorithms developed for slow attacks. Hence, we choose simulation method to evaluate algorithms proposed in this thesis.

2.4.2 Measures

Metrics, ROC Analysis and Complexity & Delay Comparison are some common measures used to evaluate IDDs.

2.4.2.1 Metrics

Four metrics are commonly used: detection rate, false positive rate, true negative rate and false negative rate (Delooze & Kalita, 2006). In addition, effectiveness and efficiency are introduced by (Kayacik *et al.*, 2005). Effectiveness is the ratio of detected attacks (true positives) to all attacks (true positives plus false negatives). Efficiency is defined as the ratio of the number of identified attacks (true positives) to all cases flagged as attacks attepmts (true positives plus false positives) (Bhuyan *et al.*, 2011).

2.4.2.2 ROC analysis

Receiver Operating Characteristics (ROC) curve has false positive rate on its X-axis and true positive rate on its Y-axis. It was introduced by Lincoln Lab for evaluation of anomaly based detection systems, currently being used by some other researches (KDD, 1999). The detection system must return a likelihood score between 0 and 1, when it detects an intrusion in order for the ROC curve to provide meaningful results. The ROC curve is used to determine how well the overall system performs, to determine the appropriate threshold values given

acceptable true and false positive rates, and to compare different detection systems (Bhuyan *et al.*, 2011).

2.4.2.3 Complexity & delay comparison

The training time, classification time and training & run-time memory requirements taken by anomaly detectors are computed using tools like HPROF[1] (Spafford & Kumar, 1994). A delay value is listed if an attack is not detected altogether. These measures indicates the performance of anomaly detection in terms of computational complexity rather than the accuracy of an anomaly detector. However, many researches very often use the evaluation metric in section 2.4.2.1 to establish their works than using any other measures (Bhuyan *et al.*, 2011).

2.5 Conclusion

Number of methods have been proposed during last three decades for computer system monitoring. Most of them focus on monitoring for typical quick attacks lasting within short period of time (few seconds or minutes). Hence such methods not suitable for monitoring slow activities. Signature based detection methods cannot be employed in slow activity monitoring due to current computational constraints of monitoring devices. The key challenge for anomaly detection in this domain is the huge volume of data, and therefore more computationally efficient (lightweight) methods are needed. Since the data amounts to millions of data objects, a few percent of false alarms can make analysis overwhelming for an analyst. Therefore false alarms need to be reduced as much as possible. Evaluation of a novel algorithm is a critical challenge in this domain. Some benchmark datasets are available, but not suitable for slow activity monitoring due to number of limitations.

All in all monitoring for slow activities in computer networks poses a set of unique challenges:

- The monitoring techniques are required to operate in an online manner.

[1]HPROF is a tool capable of presenting CPU usage, heap allocation statistics, and monitor contention profiles.

Due to the severe resource constraints in network devices, the monitoring method needs to be lightweight.

- The data is distributed, and collected in a distributed fashion from various devices at various points. Hence a distributed data analysis technique or an information fusion technique needs to be employed.

- The presence of noise in the data collected makes the analysis more challenging, and hence distinguish between interesting events and unwanted noise is required.

Chapter 3

A Bayesian framework for monitoring

This chapter presents a Bayesian approach as a method for information fusion to address effective monitoring of slow suspicious activities, and examines the effectiveness of such an approach under different network conditions. Theoretical account of the approach and detailed discussions of experimental results are included. The main contribution of this chapter is a novel algorithm for slow activity monitoring.

3.1 Introduction

Slow and low attacks can be difficult to detect. In order to detect slow activities, it is necessary to maintain a long history of what is happening in the environment. There is a processing and storage overheads involved in the state-full IDSs, as maintaining state information of every packet and comparisons between current packets and previous all packets are needed (Krishnamurthy & Sen, 2001). Most systems cannot keep enough event data to track across extended time intervals for this purpose due to the performance issues and computational constraints such as storage and processing limitations of monitoring devices (Krishnamurthy & Sen, 2001; Vallentin *et al.*, 2007; Vasiliadis *et al.*, 2011). As a result the scarcity of attack data within a short period of time allows a slow attacker to go undetected

hiding her attempts in the background noise.

This chapter proposes *profiling*, through information fusion and accumulation, as a possible method for data reduction in slow activity monitoring. It reduces the sheer volume of data gathered from various sources into a single profile score, and solves the critical issue of *maintaining long term states of systems while managing data volumes* (CSIEM, 2013). The primary objective of the proposed approach is to identify nodes for further investigation. The series of experiments presented in this chapter examine the effectiveness of such an approach under different parameters including multiple attack sources, traffic volume and subnet size.

3.2 Related work

Our work is inspired by Chivers *et al.* (2009, 2013) who provide a scalable solution to identify suspicious slow insider activities in a Bayesian framework. They profile all nodes (i.e. origin of activities) by the time and distinguish between anomalous and normal behaviours by setting a predefined control (base line). If the node score function of a particular node deviates from a predefined control then the causal node is identified as a slow suspicious node. This decision criteria is a major limitation of their approach. When there are more than one attackers in a subnet, especially with a higher variation of node behaviours, that decision criteria is not effective. It can be affected by even dimensions of the drawing canvas as any kind of standardisations is not been used. Moreover setting a predefined baseline for node behaviours is very difficult. In a network scenario normal behaviour keeps evolving by the time and a current notion of normal behaviour might not be sufficiently representative in the future. Hence, thresholds too need to evolve according to the situation and current state of the network.

Kandias *et al.* (2010) propose a model to integrate user's technological traits with data obtained from psychometric tests. In (Kandias *et al.*, 2010; Bradford *et al.*, 2004), users are profiled according to their behaviour and use that information to identify users who warrant further investigation. This is highly subjective, organizational dependent and cannot be used to profile non human actors. Basu *et al.* (2001) propose an approach which uses connection based windows to detect

low profile attacks with a confidence measure while Streilein *et al.* (2002) use multiple neural network classifiers to detect stealthy probes. Evidence accumulation as a means of detecting slow activities is proposed in (Heberlein, 2002).

(Barbara *et al.*, 2001; Sebyala *et al.*, 2002; Valdes & Skinner, 2000; Ye *et al.*, 2000; Bronstein *et al.*, 2001) use naive Bayesian networks for network intrusion detection which estimate the posterior probability of observing a class label given a test data instance. Maximum posterior is chosen as the predicted class for a given test data instance. The likelihood of observing the test instance given a class and the prior on the class probabilities are estimated from training data sets. Such techniques assume independence between different attributes. The complex Bayesian networks proposed in (Siaterlis & Maglaris, 2004; Janakiram *et al.*, 2006; Das & Schneider, 2007) capture conditional dependencies between different attributes.

Davidoff & Ham (2012) claim flow record analysis techniques are statistical in nature, but exceptionally powerful and very useful in a slow attack environment where full content captures are limited by the amount of disk space and processing power. The purpose of flow record collection is to store a summary of information about traffic flows across a network and to analyse it to find any kind of network issues which can be operational, accountable or security. Existing tools (Argus, 2012; Damien Miller, 2012; CERT Network Situational Awareness Team, 2012; ProQueSys, 2012), some switches (Cisco catalyst), routers and firewalls (current revisions of Cisco) support flow record generation and exportation (Davidoff & Ham, 2012). Our approach is different from existing flow record analysis techniques because of the Bayesian framework and evidence accumulation. Existing systems consider flow record elements such as IP addresses, ports and flags, protocols, date and time, and data amount as separate parameters without any kind of data fusion.

All above approaches, except (Chivers *et al.*, 2009, 2013; Heberlein, 2002), require storage of large volumes of event data for analysis. Systems that try to model the behaviour of individuals or protocols are forced to retain large amounts of data which limits their scalability. The work presented in this chapter is different from Chivers *et al.* (2009, 2013)'s work from the decision criteria consisting of both horizontal and vertical analysis, as well as from the way defining

the hypothesis. Heberlein (2002)'s work differs from this work as it uses counting algorithm instead of Bayesian approach and also in its decision criteria. Chivers *et al.* (2009, 2013) demonstrate Bayesian approach is superior to the counting algorithm. Since the aim of this chapter is not only just proposing an efficient monitoring algorithm but also an investigation of its effectiveness under different conditions, work presented in this chapter is novel.

3.3 Methodology

We propose an incremental approach which updates normal node profiles dynamically based on changes in network traffic. If some aberrant changes happen in network traffic over the time it should be reflected in profiles as well is the underlying assumption. Then based on profile scores it is possible to raise a suspicious activity subject to the above assumption. We look at the main problem as two sub problems: *profiling* and *analysis*. It enables us to study each part separately and reduce the complexity.

3.3.1 Motivation and source

Prior to seeking methods for profiling and analysis, it should be noted that there are two types of uncertainties: *motivation* and *source* of each event of interest. These uncertainties need to be acknowledged in profiling and analysis. Ignoring them could be resulted to more false alarms in the monitoring process.

3.3.1.1 Motivation uncertainty

The motivation behind a network event is not always easy to judge. Some suspicious events[1][2][3][4] can be appear as part of an attack as well as can originate from

[1] A major router failure could generate many ICMP unreachable messages while some computer worms (e.g.CodeRed and Nimda) generate the same in active probing process.

[2] An alert of multiple login failures could result from a forgotten password as well as could be a result of a password guessing attack.

[3] An execution of cmd.exe could be a part of malicious attempt or a legitimate as it is frequently used by malicious programs to execute commands while it is also frequently used by legitimate users during their normal day-to-day operations.

[4] An attempt to access the file robots.txt (see Appendix A for more details).

normal network activities. By just looking at alerts generated by such an event we cannot simply judge its cause. Other contextual information can be used to narrow down the meaning of such an event (Davidoff & Ham, 2012). For example, suspicious port scanning activity may have the following characteristics: a single source address, one or more destination addresses, and target port numbers increasing incrementally. When fingerprinting such traffic, we examine multiple elements (multivariate) and develop a hypothesis for the cause of behaviour on that basis.

The environment is noisy due to underlying uncertainty. In a deterministic system, the states of a dynamic process are observed and tracked without noise. But in a noisy environment such observations are often polluted by underlying noise. Therefore adopting a probabilistic approach is useful to acknowledge that uncertainty. This is because probabilistic approaches perform well in noisy environments than deterministic approaches (Jiang & Cybenko, 2004; Chivers *et al.*, 2013). A multivariate version of Bayes' formula is used for this purpose.

3.3.1.2 Source uncertainty

There is no guarantee on visible source of an event to be the true source. As mentioned in (De Tangil Rotaeche *et al.*, 2010), to remain anonymous, an attacker attempts to either disguise the elements that characterise the attack or hide the source. The localisation process becomes evermore difficult when the attacker employs various proxy methods (e.g. Generic port routing, HTTP, Socks, IRC etc) and zombie (e.g. bots) nodes. Manipulation of TCP/IP elements (e.g. IP Spoofing), using relay or random routing (e.g. Tor networks, Crowds, Freenet systems etc) approaches can help an attacker protecting her location. Proliferation of weakly encrypted wireless networks could also help an attacker getting anonymous locations.

3.3.2 Profiling

Profiling is the method for evidence fusion across space and time updating node profiles dynamically based on changes in evidence. Simply put, we compute a suspicion score for each node in the system and that score is updated as time

progresses.

3.3.2.1 Evidence fusion

Security events must be analysed from as many sources as possible in order to assess threat and formulate appropriate response. According to Drew (n.d), deploying and analysing a single device in an effort to maintain situational awareness with respect to the state of security within a network is the computerised version of tunnel vision. Extraordinary levels of security awareness can be attained in an organization's network by simply listening to what its all devices are telling you. Note that proposed profiling technique in this thesis fuses information gathered from different sources into a single score.

3.3.2.2 The Bayesian paradigm

The posterior probability of the hypothesis H_k given that E is given by the well-known Bayes formula:

$$p(H_k/E) = \frac{p\,(E/H_k)\,.p(H_k)}{p(E)} \tag{3.1}$$

Hypothesis for the monitoring algorithm is built as follows. Let H_1 and H_2 be two possible states of a node in computer network and define H_1 - the node acts as an attacker and H_2 - the node does not act as an attacker. Then H_1 and H_2 are mutually exclusive and exhaustive states. P(H_1) expresses the prior belief, in term of probability, that the node is in state H_1 in absence of any other knowledge. Once we obtain more knowledge on our proposition H_1 through multiple information sources (observation spaces), in form of evidence E={e_1,e_2,e_3,...,e_m}, our belief can be expressed as a conditional probability $p(H_1/E)$. Using the Bayes' theorem in Equation 3.1:

$$p(H_1/E) = \frac{p\,(E/H_1)\,.p(H_1)}{p(E)} \tag{3.2}$$

$P(E)$ is the probability of producing suspicious events, but on its own is difficult to calculate. This can be avoided by using the law of total probability

and reformatted Equation 3.2 as:

$$p(H_1/E) = \frac{p\,(E/H_1)\,.p(H_1)}{\sum_{i=1}^{2} p(E/H_i).p(H_i)} \tag{3.3}$$

Assuming statistical independence between information sources:

$$p(H_1/E) = \frac{\prod_{j=1}^{m} p(e_j/H_1).p(H_1)}{\sum_{i=1}^{2}\prod_{j=1}^{m} p(e_j/H_i).p(H_i)} \tag{3.4}$$

Once E is observed, to calculate the *posterior probability* $p(H_1/E)$ that the node is in state H_1 given E it is necessary to estimate:

- the *likelihood* of the event e_j given the hypothesis H_i , i.e. $p(e_j/H_i)$ and,

- the *prior probability* $p(H_i)$.

Assuming that we know the prior and likelihoods, it is possible to use Equation 3.4 to combine the evidence from multiple sources to a single value (posterior probability which is known as node score) which describes our belief during a short observation period that the node is an attacker given E. Brynielsson *et al.* (2013) is one of the most recent works following a similar idea for detecting lone wolf terrorists.

3.3.2.3 Statistical independence

In probability theory and statistics, two events are statistically (stochastically) independent if the occurrence of one does not affect the *probability* of the other. Furthermore, two (or more) random variables are independent if the realization of one does not affect the probability distribution of the others, and are identically distributed (i.i.d.) if each random variable has the same probability distribution as the others and all are mutually independent (Clauset, 2011). Independent events are not affected by other events. To give some intuition about what this means, taking coloured marbles from a bag without replacements is a series of

dependent events as there are less marbles left in the bag for the next event, so that the probabilities are changed. With replacements such events become independent.

In a loose usage of the above definition, possible sources of information to our monitoring algorithm are outputs of signature based IDSs, anomaly detection components, antivirals, file integrity checkers, SNMP-based network monitoring systems, information from L3 switches, etc. Hence the assumption on statistical independence in Equation 3.4 is reasonable as we propose to use distinct types of information sources which operate independently. In general, an alert of a suspicious event from one source does not affect the probability of alerting the same (or different) event from the other source. For example, if there are s suspicious packets in a traffic collection consists of T packets, probability of alerting a suspicious event at SNMP-based network monitoring systems (i.e. $\frac{s}{T}$) is not affected by an alert of unusual file access generated in the file integrity checker. They work statistical independently. In practice, a good Security Information Event Management (SIEM) deployment aggregates a number of solutions from many independent vendors (CSIEM, 2013). Even if we deploy identical multiple sensors of an IDS (e.g. CISCO IDS) in different locations in the same network, this assumption is held as long as they work independently. Usually, in multiple sensor deployments each sensor independently reports visibility of suspicious events. However, when using Intrusion Prevention Systems (IPS) this might not be held as such systems block/discard the causal traffic for such events on the spot.

3.3.3 Analysis

If attacker activity pattern is sufficiently reflected by profiles then detecting anomalous profiles would be sufficient to identify attackers. Hence detecting anomalous profiles in a given set of node profiles is the analysis. When there is an attacker who violates legitimate user's activity patterns, the probability that the attacker's activity is detected as anomalous should be high (Bhuyan *et al.*, 2011).

3.3.3.1 Anomaly detection

An anomaly is an irregularity of data patterns or points which deviates from expected or normal states. The problem of detecting patterns (points) in data that do not conform to expected behaviour is called anomaly detection. The terms anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities or contaminants are often used to refer these non-conforming patterns in different application domains (Patcha & Park, 2007; Chandola *et al.*, 2009; Lazarevic, 2013; Tanase, 2002).

The simple approach to anomaly detection is to define a region representing normal behaviour and declare any observation in the data which does not belong to this normal region as an anomaly. This simple approach is very challenging in network security monitoring due to the nature of anomalies. Consider the following:

- Defining a normal region in advance which encompasses every possible normal behaviour is very difficult. In addition, the boundary between normal and anomalous behaviour is often not precise;
- When anomalies are the result of malicious actions, the malicious adversaries often adapt themselves to make anomalous observations appear normal, thereby making the task of defining normal behaviour more difficult;
- Normal behaviour keeps evolving and a current notion of normal behaviour might not be sufficiently representative in the future;
- Availability of labelled data for training and validation of models used by anomaly detection techniques is usually a major issue; and
- Often data contains noise which tends to be similar to actual anomalies and hence is difficult to distinguish and remove.

Due to the above challenges (Chandola *et al.*, 2009), most of the existing anomaly detection techniques solve a specific formulation of the problem, which is induced by various factors such as data types and types of anomalies of interested, and encourage unsupervised anomaly detection techniques. Various concepts from diverse disciplines such as statistics, machine learning, data mining,

information theory, spectral theory, etc are adopted in specific problem formulations and providing solutions. We use a *statistical method* for detecting anomalous profiles.

3.3.3.2 Statistical anomaly detection

An anomaly is an observation in a dataset which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed for that dataset is the underlying principle of any statistical anomaly detection technique (Anscombe & Guttman, 1960). Such techniques are based on the key assumption that normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model (Chandola *et al.*, 2009). Subject to the above two key concepts, we propose to use following two types of techniques: *Peer analysis* and *Discord analysis* to detect anomalies in a given set of node profiles and identify slow suspicious activities.

3.3.3.3 Peer analysis

The critical challenge in slow attack monitoring is to keep information about activities of each node over an extended period of time. Aggregating posterior probability terms obtained from Equation 3.4 over the time helps to accumulate relatively weak evidence for long periods. These accumulated probability terms $\sum_t p(H_1/E)$ (t is time), known as *node scores*, can be used as a measurement of the level of suspicion for a given node at any given time; subject to the assumption that if an attacker's activity pattern is sufficiently reflected by profiles, then detecting anomalous profiles would be sufficient to identify attacker. Algorithm 1 presents node score calculation procedure for any given time. Its implementation using R Language can be found in Appendix B.1.

A given set of node profiles, e.g. profiles corresponding to a network, is a univariate data set. Hence it is possible to use the univariate version of Grubb's test (GRUBBS, 1969) (maximum normed residual test) to detect anomalous points in the set, subject to the assumption that *normal* node profiles in a given set follow an unknown Gaussian distribution (Guo, 2011; Hagen *et al.*, 2007).

Λ: A set of suspicious events;
Ω: A set of node profiles;
input : Current set Ω
output: Updated set Ω
if Λ *is non empty* **then**
    Compute the posterior probability $p(H_1/E)$;
    foreach *node score* $\omega \in \Omega$ **do**
        if *node corresponded to* ω *caused* Λ **then**
            $\omega = \omega + p(H_1/E)$;
        end
    end
end

Algorithm 1: Node score calculation for peer analysis.

Of course, the set-up where we have the distribution is very well a mixture of Gaussians. Testing of our hypothesis for any given time is a Bernoulli trial. Accumulated Bernoulli trials makes a Poisson binomial distribution (Hodges & Cam, 1960; Le Cam, 1960; Daskalakis *et al.*, 2012) which can be approximated by a Normal distribution. For each profile score ω, its z score is computed as $z = \frac{\omega - \bar{\omega}}{s}$; where $\bar{\omega}$ and s are mean and standard deviation of data set. A test instance is declared to be anomalous at significance level α if:

$$z \geq T = \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/N,N-2}}{N-2+t^2_{\alpha/N,N-2}}} \tag{3.5}$$

where N is the number of profile points in the set, and $t_{\alpha/N,N-2}$ is the value taken by a t-distribution (one tailed test) at the significance level of $\frac{\alpha}{N}$. The α reflects the confidence associated with the threshold and indirectly controls the number of profiles declared as anomalous (Chandola *et al.*, 2009). Note that the threshold T adjusts itself according to current state of a network. This is a vertical analysis to detect one's aberrant behaviour with respect to her peers, hence we call it *peer analysis*. Looking at one's aberrant behaviour within similar peer groups (e.g. same user types, departments, job roles, etc.) would give better results in terms of false alarms than setting a universal baseline (Eldardiry *et al.*, 2013; Berk *et al.*, 2012). Hence first classifying similar nodes into peer groups, based on behaviour related attributes/features, and then applying the monitoring

algorithm is recommended as it accounts for regular variations over peer groups. Investigation for a suitable classification algorithms for this task depends on the nature of the target network and its attributes.

3.3.3.4 Discord analysis

When a slow attack is progressing, malicious activities are occurring according to an on-off pattern in time. As a result, lack of agreement or harmony between points in the profile sequence of a given node can occur in a similar or different on-off fashion. This type of anomalies are known as discords (Yankov *et al.*, 2008; Bu *et al.*, 2007; Yankov *et al.*, 2007; Fu *et al.*, 2006). In a slow attack environment, discords are random time context and peer analysis technique itself is not sufficient to detect them if the progression rate of malicious activities is far lower than the similar innocent activities. This is illustrated in Section 3.5.3

The objective of this analysis is to detect sub-sequences within a given sequence of profiles which is anomalous with respect to the rest of the sequence. Problem formulation occurs in time-series data sets where data is in the form of a long sequence and contains regions that are anomalous. A generic technique in this category can be described as follows. A model is learned from the data in the recent history, which can predict the expected behaviour with respect to the learned context. If the expected behaviour significantly deviates from the observed behaviour, an anomaly is declared. A simple example of this generic technique is regression in which the contextual attributes can be used to predict the behavioural attribute by fitting a regression line on the data (Chandola *et al.*, 2009). However, in real-life research and practice, patterns of the data are unclear, individual observations involve considerable error (random shock), and we still need not only to uncover the hidden patterns in the data but also generate forecasts. Simple regression is not sufficient to model such datasets due to this complexity.

Any time series can be described in terms of regular and irregular components (random shocks) (StatSoft, 2014). An ARIMA models has three parts, although not all parts are always necessary, to describe the effects of each component. These three parts are auto-regression (AR), integration (I) and moving average

(MA). The AR part describes the dependency of the observed value on some linear (or non-linear) combination of previously observed values, which is defined to a maximum lag denoted by p, plus a random error term. The MA part describes the observed value as a random error term plus some linear combination of previous random error terms up to a maximum lag denoted by q. To analyse the time series using AR and MA terms, it requires that all of the observations are independently identifiable. This is called stationary which can be achieved via differencing. In order to be stationary, all of regular (i.e. trend and seasonal) effects must be removed from the series so that series is left with only the irregular (random) components. The process of differencing is known as integration, and the order of differencing is denoted by d. The ARIMA methodology has confirmed its power and flexibility, and has gained enormous popularity in many areas and research practice for modelling complex time series (StatSoft, 2014). Hence, we propose to use Auto-Regressive Integrated Moving Average - $ARIMA(p, d, q)$ model. In this work model parameters p, d, q are automatically estimated by a built-in R script.

As in most other analyses, in time series analysis it is assumed that the data consists of a systematic pattern and random shock (error) which usually makes the pattern difficult to identify using simple regression. It requires that the pattern of observed time series data is identified and more or less formally described. Once the pattern is established, we can interpret and integrate it with data to validate it. Regardless of the depth of our understanding and the validity of our interpretation (theory) of the phenomenon, we can extrapolate the identified pattern to predict future events (StatSoft, 2014). In summary, the underlying idea of this analysis is that the normal behaviour of the time-series follows a defined random pattern, and a subsequence within the long sequence which does not conform to this pattern is an anomaly. In general, the purpose of this analysis is to detect one's aberrant behaviour with respect to her own behaviour regardless of her peers. We propose following method for discord analysis using ARIMA model.

At the $(t-1)^{th}$ time point, using an autoregressive integrated moving average model $ARIMA(p, d, q)$ which describes the autocorrelations in the data (Chatfield, 2003), 95% Confidence Interval (CI) for the t^{th} profile score is predicted. If

the observed profile score at time t lies outside of the predicted CI then absolute deviation of the profile score from CI is calculated. This deviation is used as a measure of non-conformity of a given profile score to the pattern of its own sequence (group norms). These deviations average out over the time to calculate the *anomaly score* for a given node. Note that this anomaly score is the average dissimilarity of profile scores with its own profile sequence of a node. This dissimilarity occurs randomly from time to time due to the deliberate intervention of the attacker. Algorithm 2 below presents the anomaly score calculation procedure while its implementation using R Language can be found in Appendix B.2.

τ: An $ARIMA(p, d, q)$ model, p, d, q are estimated by the built-in R script;
ω: 95% CI for the profile score at time t;
ϑ: observed profile score at time t;
input : Set of profile scores of a given node
output: An anomaly score
foreach *point in the time line* **do**
    Fit τ using n number of previous profile points;
    Compute ω;
    if $\vartheta \geq upperlimit(\omega)$ *or* $\vartheta \leq lowerlimit(\omega)$ **then**
        Deviation=Deviation+absolute(ϑ - ω);
        Anomaly score=Deviation/(number of cut-off points so far);
    end
end

Algorithm 2: Anomaly score calculation for discord analysis.

Due to the lack of training data available, we propose to run above algorithm in an unsupervised mode. The length of the ARIMA model (i.e. n - number of previous points to be used) is critical as containing anomalous regions in input sequence makes difficult of creating robust model of normalcy. Note that keeping the length of the ARIMA model less than the minimum of time gaps between two consecutive attack activities will give better results. However since the time gap between two consecutive attack activities is unknown in advance, using a smaller observation window (i.e. slicing whole observation period into many smaller parts as much as possible) to generate short time profiles would be the better. Marceau (2000) applies a similar technique to detect system call intrusions using Finite

state automata (FSA). He uses FSA to predict the next event of a sequence based on the previous n events, and if the observed value significantly deviates from the predicted value an anomaly is declared.

It should be noted that both above analyses are based on parametric techniques which assume the knowledge of underlying distribution and estimate the parameters from the given data. However, in case of distribution is unknown, it is possible to use non parametric techniques to determine the model of distribution from given data first and then to use parametric method to detect anomalies as described above.

3.4 Experiment

As mentioned in section 2.4.1, in the domain of intrusion detection, evaluation of novel algorithms against real time network data is difficult (Bhuyan *et al.*, 2011; Kayacik *et al.*, 2005; KDD, 1999). Evaluation even against an offline real network data set is a challenge if the focus is on slow attacks. Because current computational constraints, specially storage requirements, does not permit to capture and store all data even in a average network for longer times to produce such datasets. To the best of our knowledge there is no any prominent data set available for this task focusing on slow attacks. Note that existing IDS evaluation datasets, e.g. DARPA -the most widely employed data set and ISCX2012-a modern dataset, for typical attacks also have simulated both attack and normal traffic in real world networks to produce datasets (see section 2.4.1 for more details about evaluation datasets). Because it is a real challenge to find data produced by actual attackers. Therefore to assess the feasibility of our proposed approach a series of experiments were conducted simulating slow suspicious activities in simulated networks. Simulating such activities in a real network certainly gives more realistic conditions than in a simulated network. However practical constraints of the project keep away us using a real world network for this purpose at this stage.

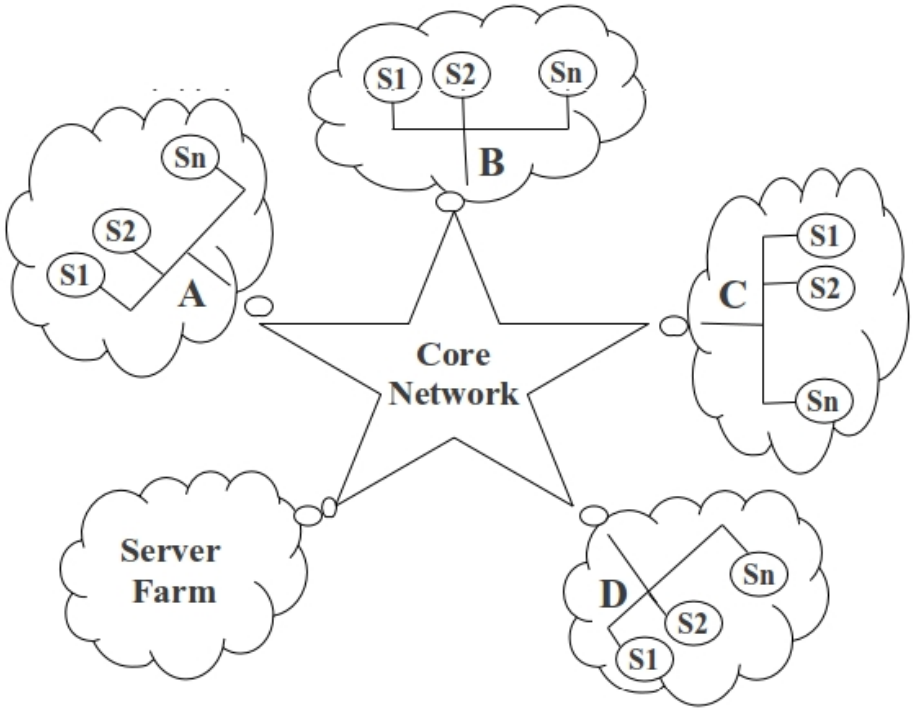

Figure 3.1: Network topology used for experiments.

3.4.1 Experimental design

Network Simulator 3 (*NS3*) (Riley & Henderson, 2010) was used to build the network topology given in Figure 3.1 and to generate the traffic patterns of interest. Poison arrival model, with inter-arrival time gap between two consecutive events as an exponential, was assumed. Each simulation was run for a reasonable period of time (over one millions events) to ensure that enough traffic was generated. The C++ codes written in NS3 for simulations of our experiments presented in this thesis can be found in Appendix B.3.

3.4.1.1 Network topology

Figure 3.1 shows the network topology used for our experiments. A total of 2,122 nodes were distributed among four networks labelled as A (99 nodes), B (400 nodes), C (800) and D (800 nodes). In addition, a network dedicated to a server farm was simulated with 23 nodes.

3.4.1.2 Suspicious events

Attackers were located one in each of the three different subnets. Anomalous traffic, by means of unusual port numbers like 32769, 33333 etc, was generated in addition to generating usual traffic within and between subnets and to external networks. To simulate uncertain events like forgotten passwords, suspicious events were generated by normal nodes as well, but at different rates. If λ_a, λ_n are mean rates of generating suspicious events (where we only generate a subset of flow data elements including source and destination address and port numbers, and where suspicious activity is judged by unexpected port numbers) by attacker and normal nodes (i.e. the noise) respectively, we ensured maintaining $\lambda_a \in \lambda_n \pm 3\sqrt{\lambda_n}$ and $\lambda_n (\leq 0.1)$ sufficiently smaller for all our experiments to characterise slow suspicious activities which aim at staying beneath the threshold of detection and hiding behind the background noise. The idea to use the above relationship for generating attacker activities was to keep them within the *normality range* of innocent activities (i.e. background noise). $\sqrt{\lambda_n}$ is the standard deviation of rates of suspicious events generated by normal nodes.

3.4.1.3 Parameter estimation

Prior probabilities and Likelihoods are assigned as follows.

$$p(H_1) = \frac{1}{2} = 0.5 \tag{3.6}$$

Equation 3.6 believes there is a 50% chance for a given node to be a slow attacker. However, this is not the case in many situations. In networks, one node may have a higher prior belief of being suspicious than another. Since prior probabilities are based on previous experiences, $p(H_1)$ can be judged based on information gathered from contextual analysis. However if there is no basis to distinguish between nodes or groups of nodes equally likely (i.e.same probability of occurring) can be assumed.

For the experiment presented in this chapter, we followed the equally likely assumption initially, and prior probabilities were assigned as in equation 3.6. Then the posterior probability of a given node at time $t-1$ is used as the prior of the same node at time t when time is progressing. This lets prior probabilities

to adjust itself dynamically, according to suspicious evidence observed over time.

$$p(e_j/H_1) = k \tag{3.7}$$

Equation 3.7 expresses the likelihood of producing event e_j by a subverted node. For the purpose of demonstration we assigned arbitrary, but different, values ($\leq$1) for k to distinguish different type of events (e_j) produced for the simulation. Likelihoods for real world implementation can be estimated as follows.

If e_j is an event resulting from a certain type of known attack (e.g. a UDP scan or LAND[1] attack), then k can be assigned to one. However, k cannot always be one, as described in section 3.3.1.1, as there are some suspicious events (e.g. an alert of multiple login failures) that can be part of an attack signature as well as originate from normal network activities. The question is how to estimate $p(e_j/H_1)$, i.e. the true positives ratio, if e_j becomes such an observation. One possible solution would be to use existing IDS evaluation datasets such as ISCX 2012 (Shiravi *et al.*, 2012) or DARPA (McHugh, 2000) to estimate true positives ratios. Estimating likelihoods for real world implementation is left as a future work

According to Chivers *et al.* (2009), in some cases, the historical rate of occurrences of certain attacks is known and can be used to estimate the likelihood that certain events derive from such attacks or it may be sufficient to quantify these frequencies by an expert in a similar way to estimating risk likelihoods to an accuracy of an order of magnitude. Note that Chivers *et al.*'s claim is completely theoretical as it follows the *Subjectivist*[2] interpretation of probability theory (GeNIe, n.d). According to Davidoff & Ham (2012), the biggest challenge is the absence of large publicly available data sets for research and comparisons, but within an organization it is entirely possible to empirically analyse day-to-day traffic and build statistical models of normal behaviour.

[1] A Denial of Service (DoS) attack which sets the source and destination information of a TCP segment to be the same. A vulnerable machine will crash or freeze due to the packet being repeatedly processed by the TCP stack.

[2] There are three fundamental interpretations of probability: Frequentist, Propensity and Subjectivist. In Subjectivist, probability of an event is subjective to personal measure of the belief in that event is occurring.

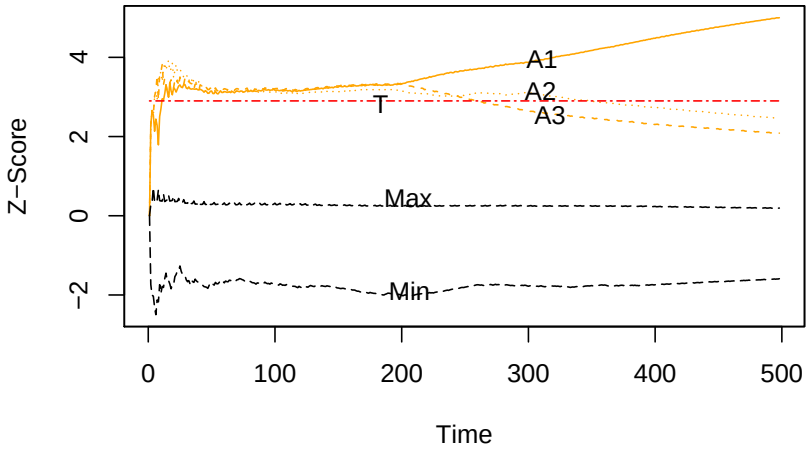

Figure 3.2: Z- Score graphs are sensitive to node behaviour.

3.5 Results

In this section experimental results are presented. We use graphical forms (e.g. Z-Score graphs) to present information. As mentioned in section 2.3.2 visualisation helps to quickly recognise patterns in data.

3.5.1 Node behaviour

First of all, to investigate whether proposed Z-score graphs reflect the behaviour of nodes, three attacker nodes were located in a 50 size subnet in network D in Figure 3.1. All others were innocent. Two out of three attackers stopped their attack activities at 200 and 300 time points respectively. Figure 3.2 presents the outcome, where $A1$, $A2$ and $A3$ are attacker nodes while Min and Max are the minimum and maximum Z-scores of normal nodes. T is the Grubbs' critical value (threshold) for a single outlier. If an attacker node changed its behaviour, the corresponding z-score graph (see $A2$ and $A3$ in Figure 3.2) responses to that behaviour by changing its direction.

3.5.2 Peer analysis outcomes

Our peer analysis technique was tested against 24 test cases as shown in Table 3.1. Test cases are different from each other by the subnet size and the number of attackers as these two parameters (i.e. peer information) are important and can affect detection. Cases 1, 7, 13 and 20 are the best case scenarios while cases 6, 12, 18 and 24 are the worst case scenarios under each subnet size. Defining each test case was completely arbitrary, and there was not specific reason to choose these test cases except best and worst cases. Our approach was capable of detecting slow attackers in all cases, except worst cases. However, only one case (Case 16) is presented here. In this case, four slow attackers were located in a hundred size subnet. At each time point, node profiles were calculated as described in section 3.3.2 and converted to Z-scores as described in section 3.3.3.3 and plotted against time as in Figure 3.3. In Figure 3.3, nodes corresponing to $A1$, $A2$, $A3$ and $A4$ denote attackers. Min and Max denote the minimum and the maximum Z-scores of normal nodes at each time point. T is the Grubbs' critical value (threshold) for a single outlier.

3.5.2.1 Aberrant peers

Aberrant node profiles A1, A2, A3 and A4 in Figure 3.3 always corresponded to the four slow attackers located in the subnet. They are above or near the threshold (T), and most importantly, there is a clear visual separation between the set of normal nodes and anomalous nodes. Hence it is possible to recognise slow suspicious activities using the proposed method.

3.5.2.2 Best and worst cases

Behaviour of the proposed approach in best and worst cases is also investigated. There were no attacks in best cases (test cases 1,7,13 and 20) while all nodes were subverted in worst cases (test cases 6,12,18 and 24). Similar graphs, as shown in Figure 3.4, were obtained in all cases. Almost all the nodes are nearly below the threshold (T), and none of nodes can be seen clearly separated from the majority. However note that Discord analysis technique is capable of detecting attackers in these cases too.

Test Case	Subnet Size	Number of Attackers
1	Twenty five nodes	No attacker
2	Twenty five nodes	One attacker
3	Twenty five nodes	Two attackers
4	Twenty five nodes	Four attackers
5	Twenty five nodes	Seven attackers
6	Twenty five nodes	All attackers
7	Fifty nodes	No attacker
8	Fifty nodes	One attacker
9	Fifty nodes	Two attackers
10	Fifty nodes	Four attackers
11	Fifty nodes	Seven attackers
12	Fifty nodes	All attackers
13	Hundred nodes	No attacker
14	Hundred nodes	One attacker
15	Hundred nodes	Two attackers
16	Hundred nodes	Four attackers
17	Hundred nodes	Seven attackers
18	Hundred nodes	All attackers
19	Two hundred and fifty nodes	No attacker
20	Two hundred and fifty nodes	One attacker
21	Two hundred and fifty nodes	Two attackers
22	Two hundred and fifty nodes	Four attackers
23	Two hundred and fifty nodes	Seven attackers
24	Two hundred and fifty nodes	All attackers

Table 3.1: Subnet size and number of attackers in each test case.

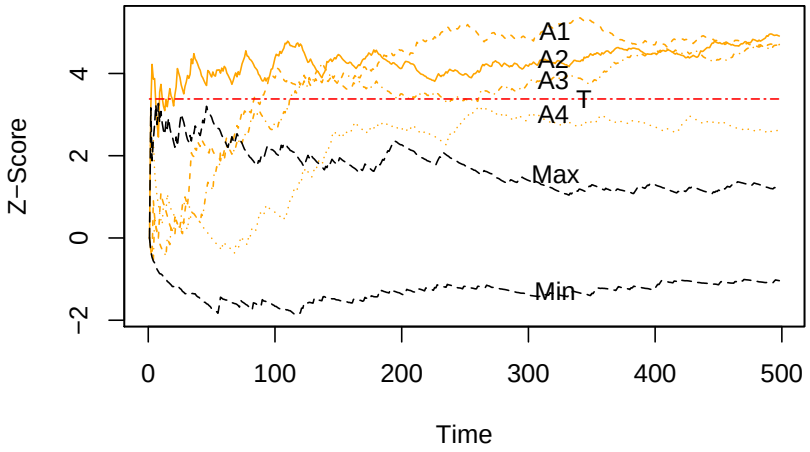

Figure 3.3: Z-Scores of node profiles for test case 16.

In a situation monitoring system depends only on peer analysis technique and has been seen similar graphs as in worst (or best) cases, it is safe to assume all nodes are subverted (instead of assuming free of attackers) and doing further investigations on one or two nodes to verify. If investigated nodes are attackers it is reasonable to consider all nodes are attackers or vice versa.

3.5.3 Discord analysis outcomes

Discord analysis technique was tested against number of test cases in Table 3.1 in addition to testing it against a special test case defined as follows. In a slow attack environment, discords are random time context and peer analysis technique itself would not be capable to detect them if the progression rates of malicious activities are far lower than the rates of similar innocent activities. Therefore a small subnet consisting of five nodes including one attacker was set-up in network D in Figure 3.1. Attacker's activity rate was decreased until observing a node score graph like in Figure 3.5 where peer analysis technique itself fails to detect the attacker. In Figure 3.5, the attacker which is denoted by red dotted line always keeps a very low profile score than all innocent nodes denoted by other

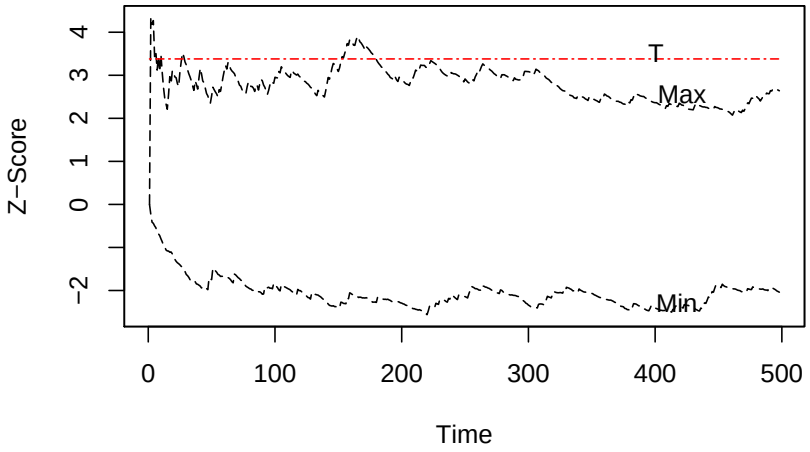

Figure 3.4: Z-Scores of node profiles for test case 7.

lines (see magnified version in Figure 3.6). As it is clearly seen in Figures 3.5 and 3.6 attacker hides behind the normal nodes, and since attacker's profile score is far lower than all normal nodes it is not detected by the peer analysis technique. Randomness of event generation can also be clearly seen from Figure 3.6.

Algorithm 2 in section 3.3.3.4 was applied for this case and observed that discord analysis is capable of detecting the attacker very well. First using an ARIMA(p, d, q) model 95% CI is predicted for each node in the network (see Figures 3.7 and 3.8 which are created for the attacker node and a normal node respectively). Then absolute deviations of the profile score from CI is calculated, for example the distance between points P1 and P2 in Figure 3.7. These deviations average out over the time to calculate the *anomaly score* for a given node and converted into Z-scores and plotted against the time line as in Figure 3.9. Twenty five previous points were used as the length of the ARIMA model. In Figure 3.9, the node corresponded to A denotes the attacker. Min and Max denote the minimum and the maximum Z-scores of anomaly scores of normal nodes at each time point. T is the Grubbs' critical value (threshold) for a single outlier. As it is obvious in Figure 3.9 attacker node is clearly distinguished from innocent nodes.

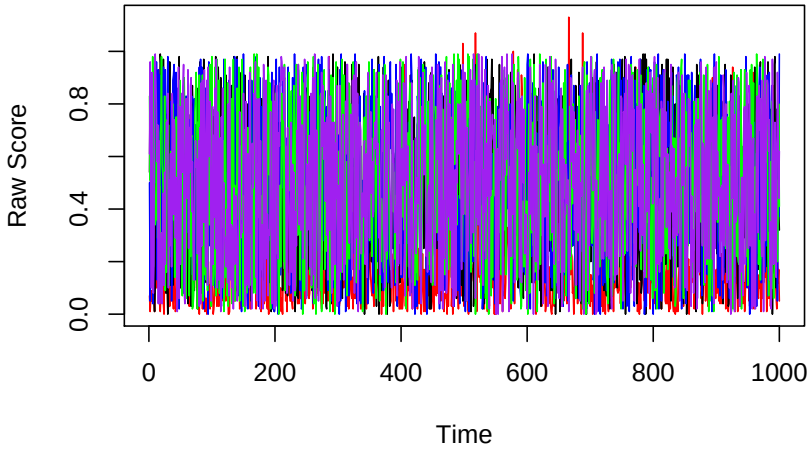

Figure 3.5: Hiding behind innocent nodes (See magnified version in Figure 3.6)
.

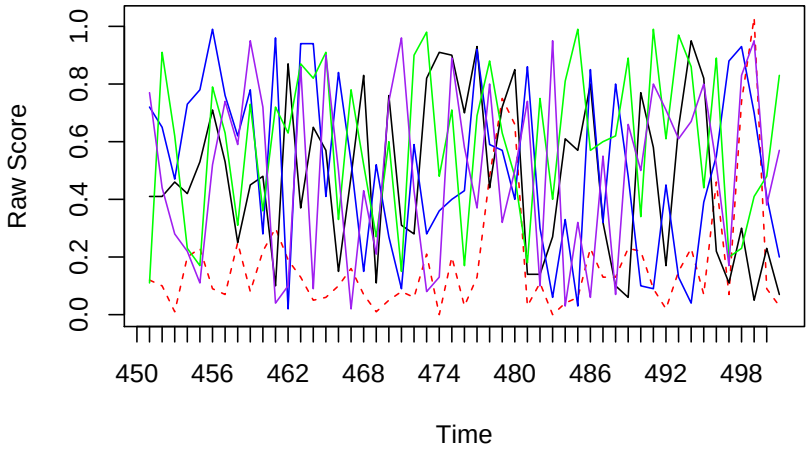

Figure 3.6: Magnified version of Figure 3.5 - red dotted line denotes the attacker, all other lines denote innocent nodes.

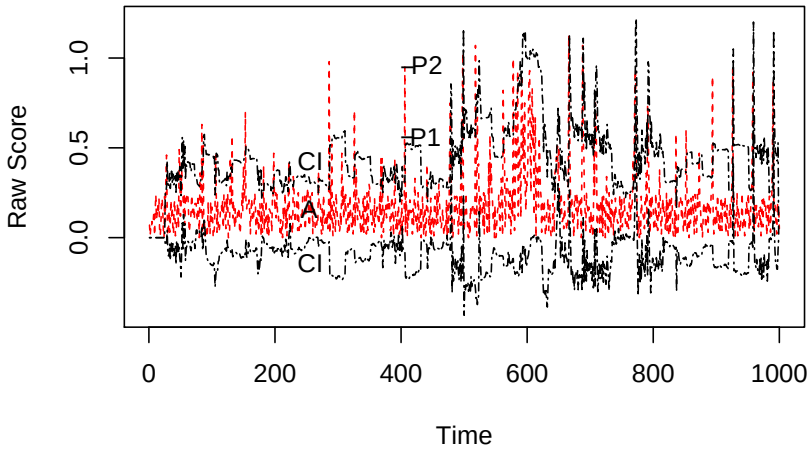

Figure 3.7: Node scores and 95% CI intervals for the attacker node. Black lines denote CIs while the red line denotes the attacker (A).

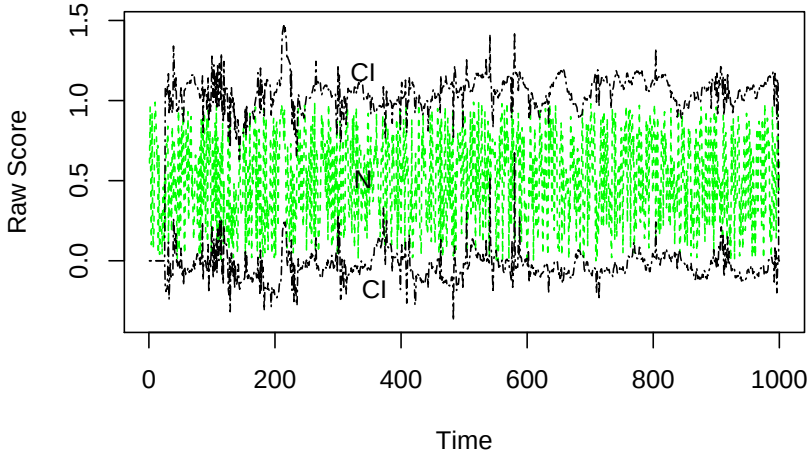

Figure 3.8: Node scores and 95% CIs for a normal node. Black lines denote CIs while the green line denotes the normal node (N).

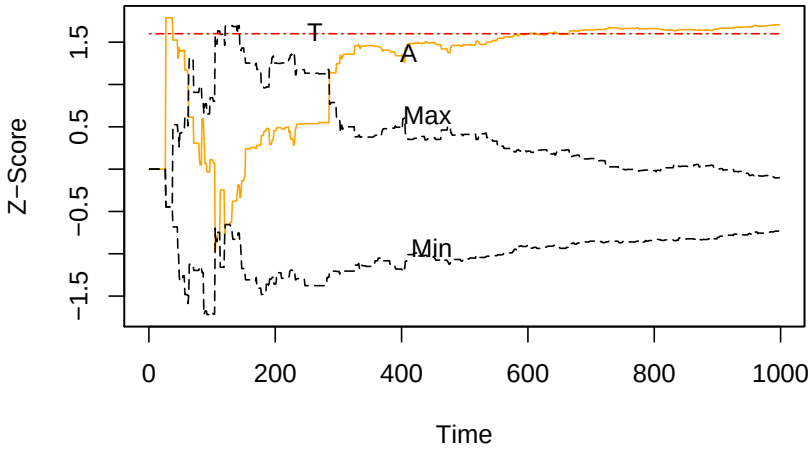

Figure 3.9: Z-Scores of anomaly scores for Discord analysis.

3.5.4 Network parameters

In this section we investigate how different network parameters: traffic volume, subnet size and number of attackers affect on monitoring of slow activities. A measure called *detection potential* is defined in this section for this analysis. That measure explains how far an attacker node is deviated from the average of normal nodes (statistical norm), and helps to compare between different network conditions.

3.5.4.1 Traffic volume

An attacker was located in a 51 size subnet of Network C and generated suspicious events. The same experiment was repeated six times by keeping all parameters unchanged, except attacker's traffic volume. If the attacker's traffic volume is V at the first time, then at each repetition the attacker's traffic volume was incremented by one time as $2V$, $3V$, ...,$7V$. For each experimental run detection potential was calculated, and standardized values of detection potentials (z-scores of deviations) are plotted as in Figure 3.10. As shown in Figure 3.11 detection potential is proportional to the traffic volume. The higher the traffic volume

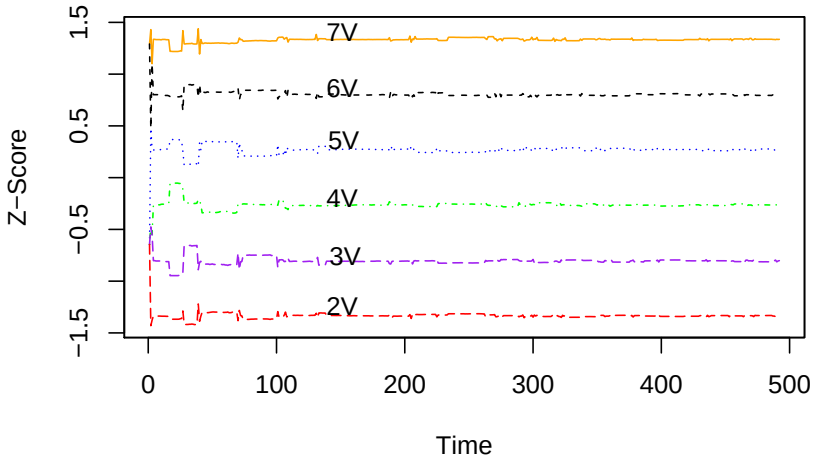

Figure 3.10: Z-Scores of deviations of cumulative node scores.

produced by an attacker is the better for her detection using the monitoring algorithm.

3.5.4.2 Subnet size

To investigate how the subnet size affects on detection, an attacker was located in a 500 size subnet and the same experiment was repeated six times by keeping all other parameters, except the subnet size, unchanged. Subnet size was changed to 400, 300, 200, 100, 50 and 25 at each experimental run and the graphs in Figures 3.12, 3.13 and 3.14 were obtained. According to graphs in Figure 3.12 and 3.13, attackers have less chance to hide behind innocent events, when the subnet size decreases. It is further reinforced by 3.14 showing that "the smaller the subnet size, the better for detection of suspicious slow activities". But in practice, it should be noted that partitioning a network into very small subnets would not be a feasible solution in sometimes, as it depends on several other factors such as resources availability and user requirements. According to the Figure 3.14, detection potential is negative exponential to the subnet size and going beyond 100 size subnet size would not make any real sense in terms of

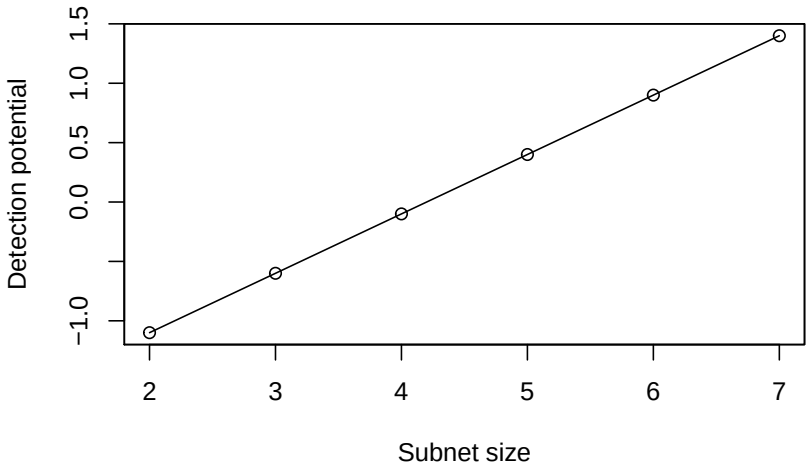

Figure 3.11: Traffic volume vs Detection potential.

detection in this case.

3.5.4.3 Number of attackers

Keeping all conditions unchanged, except number of attackers, the same experiment was repeated many times as described in table 3.1. The outcomes of only two test cases (21 and 22) are presented in Figures 3.15 and 3.16. The attacker's node score (see figures 3.15 and 3.16) is dependent on "the number of attackers in her own subnet" (compare attackers' Z-scores). It rationalises our choice to using peer analysis technique. Looking at one's aberrant behaviour with respect to her peers (i.e within the same domain, department, similar user group, region, country etc.) would give better results (in terms of lower false alarms) than defining it universally. Comparison of nodes profiles regardless of similar peer groups (e.g. subnets) would give higher false alarms. Hence, first classifying similar nodes into peer groups, based on behaviour related attributes/features, and then applying the monitoring algorithm will give better results. Investigations for suitable classification algorithms for this task is left as a future work.

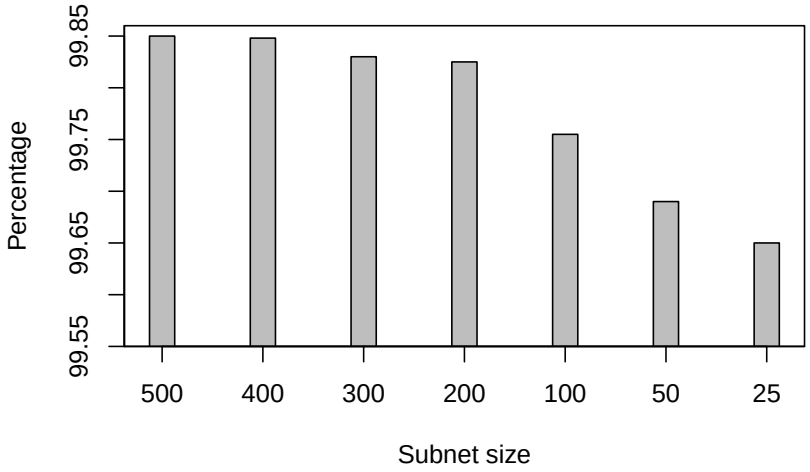

Figure 3.12: Percentages (%) of suspicious events generated by all innocents.

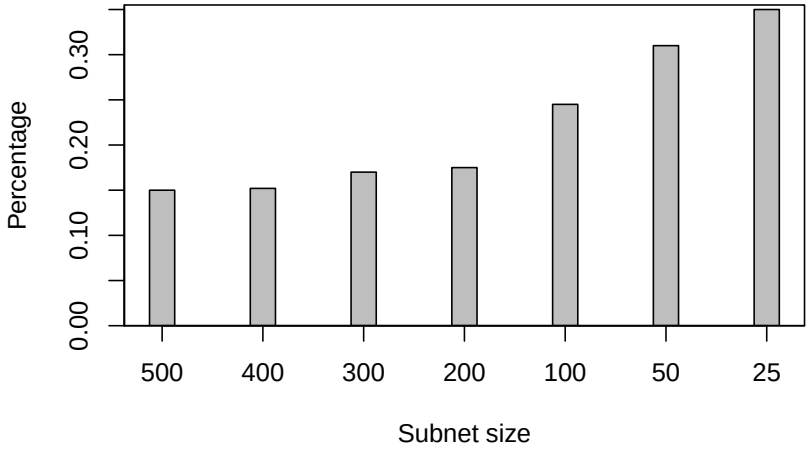

Figure 3.13: Percentages (%) of suspicious events generated by the attacker.

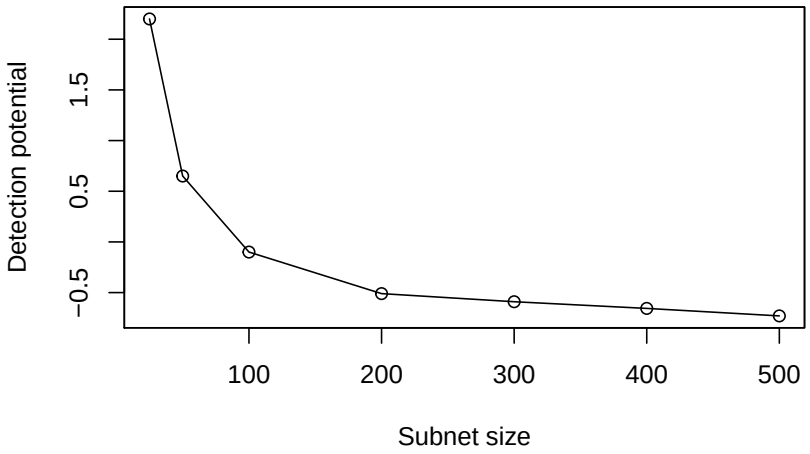

Figure 3.14: Subnet size vs Detection potential.

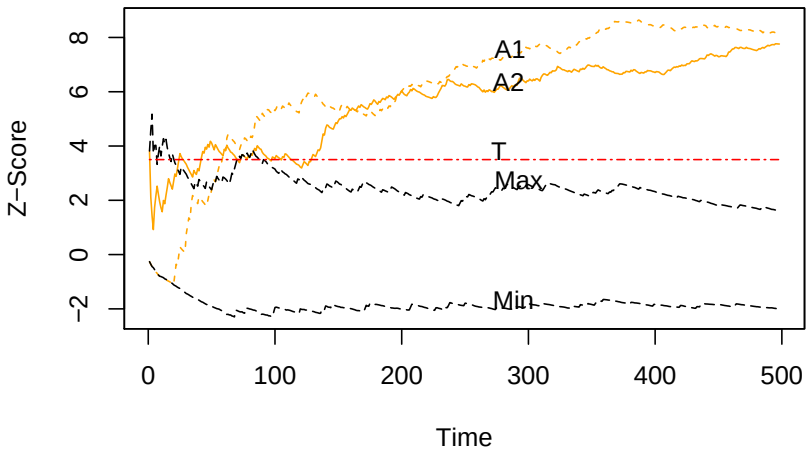

Figure 3.15: Z-Score graphs for same size subnets with different number of attackers (250 size subnet, two attackers).

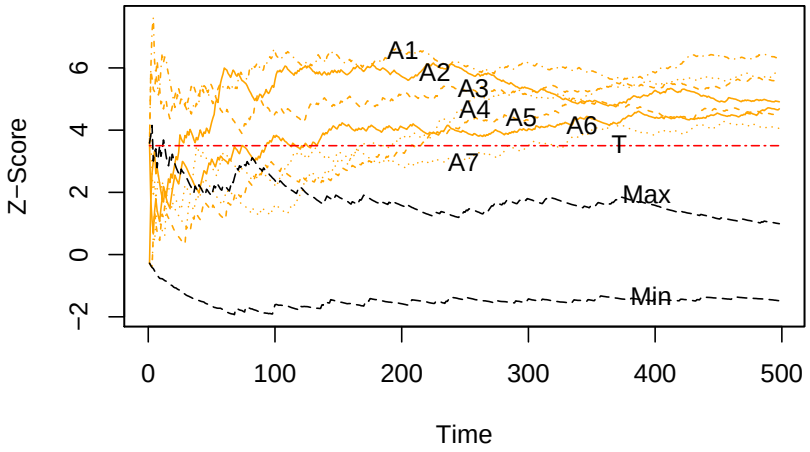

Figure 3.16: Z-Score graphs for same size subnets with different number of attackers (250 size subnet, seven attackers).

3.6 Discussion

The purpose of the proposed approach is to serve as an early warning system for slow suspicious activities over computer networks that warrant further investigations. Long-term state of a node is restricted to an incremental estimate of the probability that each node is an attacker. Node estimates are updated following every security event taking account of transient network information that is available at the time of the event. A state size (i.e. a node score) in peer analysis technique is a small multiple of the number of nodes in the network and hence its storage is feasible even for organizations with global networks. Even in the discord analysis technique using about 25 profile points (i.e. holding only 25 numbers) would be enough to fit the ARIMA model for a given node and hence is scalable in terms of storage.

However in terms of processing discord analysis technique may demand more processing power than the peer analysis technique because of its model fitting part. Therefore using one of either techniques (or even both together) would be a trade-off between a computational cost and required level of precision of

the monitoring process. Note that analysis in rest of chapters in this thesis depends on peer analysis technique only as it provided required precision for the purpose of such analyses and also due to its lightweight nature. A study of the requirements of host statistics such as CPU load, I/O operations and memory usage for proposed techniques is out of the scope of this study, and hence has not been included in the thesis. Because the aim of this study is to establish the theoretical framework for a monitoring scheme rather than detailed implementation issues.

Proposed approach in this thesis is an incremental approach which accumulates evidences by the time to detect slow activities that occur over longer periods of time. Hence our approach would not be suitable for handling situations if the progression of attack activities are rapid (i.e. typical quick attacks lasting within few seconds or minutes). Proposed methods have not been evaluated against real world network data. Because it is very hard to find useful network traces from real systems for IDS evaluations, especially, in a situation needed larger variations of network parameters during very long times. Though each technique proposed in this thesis has been tested in simulated environments, simulation environments may not necessary be representative to the exact real world networks. Therefore one possible option to evaluate the proposed approach under more realistic conditions would be using a community-lab test bed. Since our aim is at this stage to establish the conceptual framework that type of evaluation is left as a future work.

The model using to generate node profiles in this thesis is a rather simple, in fact, the multivariate version of the well known simple Bayesian formula. It was chosen for this work because of its simplicity. Attributes are conditionally independent is the simplified assumption in this version which greatly reduces the computation cost (Wasilewska, 2010). However, there can be dependences between value of attributes. To avoid this, extended versions of Bayes' theorem such as Bayesian Belief networks which provide joint conditional probability distribution can be used. Such a choice may add more precision to the monitoring process in terms of accuracy, but can demand for more computational resources of the monitoring devices (Kjaerulff, 1994; Zhang & Poole, 1994). However, investigations to find the optimal method among number of different techniques for

information fusion[1] would be interesting, but out of the scope of this thesis and interesting readers are invited to refer (Hall & McMullen, 1992; Hall & Llinas, 2001; Hall & Garga, 1999; Hall & Llinas, 1997; Hall & McMullen, 2004) for more details on different information fusion techniques including their *pros and cons*. Using Dempster-Shafer's theory of evidence towards multi sensor data fusion for DoS detection can be found in (Siaterlis & Maglaris, 2004). Following a taxonomy in (Hall & McMullen, 1992), Siaterlis & Maglaris identify Dempster's-Shafer's theorem as the promising method for their work.

3.7 Conclusion

An efficient method for slow activity monitoring and investigation of its effectiveness under different conditions have been proposed, simulated and demonstrated. Experimental outcomes and recommendations presented in this chapter provide tactical and operational principles for systematic and efficient profiling and analysis. They are particularly useful in the capacity planning stage of network design process.

Breaking down the monitoring problem into two sub problems reduces the complexity of the problem and explores ways to investigate alternative methods. Acknowledging the event uncertainty to the monitoring process reduces the false alarms and hence provides better monitoring. Our approach is a complementary to conventional intrusion detection techniques but not a replacement. Proposed approach is domain agnostic. It can be extended to use for the wider context of Cyber defence (e.g. profiling geographical locations of adversaries), and can be used also in other domains such as crime and juridical sciences.

This chapter proposed a novel algorithm for slow activity monitoring in a Bayesian framework. Theoretical account of the approach was established and detailed discussions of experimental results were included. Next chapter uses the proposed Bayesian framework to provide an anomaly based adaptive method for tracing down the sources of slow suspicious activities. Chapter 5 examines the

[1] Investigations to use some advanced methods like N-gram version of HMM, Bayesian Belief networks, Kernel Density Estimation (KDE), Dempster-Shafer theorem, Kalman Filter, Viterbi algorithm, Gi*, Evidential reasoning, Logic based fusion, Preference aggregation, Neural networks and Ontology & category theory to generate node profiles would be interesting.

feasibility of employing traffic sampling with the proposed monitoring algorithm in this chapter. A target-centric monitoring scheme is presented in Chapter 6 utilising only the destination information of activities by the algorithm.

Finally the technical findings of this chapter include:

- breaking down the monitoring (main) problem into two sub problems (profiling and analysis) which reduces the complexity of the problem and enable researchers to study/improve each part separately;

- demonstrating the feasibility of using a probabilistic, particularly a Bayesian, approach for profiling to acknowledge motivation uncertainty. It reduces possibility of occurring false alarms;

- introducing peer analysis technique consisting of a dynamic threshold which capable of adjusting itself according to the current state of the network;

- introducing discord analysis technique which capable of detecting slow suspicious activity regardless of how much is it slow.

Chapter 4

Tracing slow attackers

This chapter discusses the tracing down problem, and provides an anomaly based adaptive method for tracing down the sources of slow suspicious activities in computer networks. A theoretical account of the approach and experimental results are provided. The main contribution of this chapter is the tracing algorithm.

4.1 Introduction

Tracing down anonymous slow attackers creates number of challenges in network security. There is no guarantee on publicly visible source of an event is to be the true source. As mentioned in (De Tangil Rotaeche *et al.*, 2010) to remain anonymous the attacker attempts to either disguise the elements that characterize the attack or hide the source of its acts. The localization process becomes evermore difficult when the attacker employs various proxy methods (e.g. Generic port routing, HTTP, Socks, IRC etc) and zombie (e.g. bots) nodes. Manipulation of TCP/IP elements (e.g. IP Spoofing), using relay or random routing (e.g. Tor networks, Crowds, Freenet systems etc) approaches can help an attacker protecting her location. Proliferation of weakly encrypted wireless networks could also help an attacker getting anonymous locations.

The methodology presented in Chapter 3 is based on underlying assumption that attack activity can be attributed to a meaningful specific source or an intermediate. This assumption is not held for anonymous slow attackers and identi-

fying source of such an attack requires tracing packets back to the source hop by hop. Current approaches for tracing these attacks require the tedious continued attention and cooperation of each intermediate Internet Service Providers (ISPs). This is not always easy given the world-wide scope of present day Networks (Burch & Cheswick, 2000). Many researchers claim completely depending on information derived from single network device will do little on Cyber conflict attribution and detection due to the nature of the current Internet infrastructure (Drew, n.d). Therefore there is a need for approaches that combine technical solutions data with the information gathered from contextual analysis (combining prior belief with posterior knowledge).

This chapter presents a methodological way to trace anonymous slow activities to their approximate sources by prioritizing evidence acquisition. First, network paths from a victim to its immediate visible hops are mapped as a tree. Then each path is profiled in a Bayesian framework and highest scored path is chosen to move towards. This is repeated until the exact location of the attacker is found or the entire topology is covered. As shown in the rest of the chapter this method eliminates all but a handful of suspicious nodes that could be the source of the suspicious activity.

4.2 Related work

Parker (2010) claims that a common problem with many analysis tools and techniques today is that they are simply not designed for purposes of attribution. Beidleman (2009); Morrill (2006); Wheeler & Larsen (2003) mention attribution of Cyber activity - "knowing who is attacking you" or "determining the identity or location of an attacker or an attacker's intermediary"- is naturally a vital ingredient in any Cyber security strategy. Although current approaches are capable of alarming suspicious activities, most of them are not suitable for this information age because when computers are under attack "who" and "why" are frequently unknown (Charney, 2009; Saalbach, 2011).

Tracing back is one of the most difficult problems in network security, and a lot of research being conducted in this area (John & Sivakumar, 2009; Mitropoulos *et al.*, 2005). But deterministic packet marking and out of band approaches are

not relevant to this work. Because our approach is a probabilistic approach. As explained in Chapter 3 deterministic approaches are not capable to acknowledge motivation uncertainty and leading to more false alarms (Chivers *et al.*, 2009, 2013; Heberlein, 2002).

Flooding tests network links between routers are controlled to approximate the source in (Burch & Cheswick, 2000). Sager and Stone suggest to log packets at key routers and then to use data mining techniques to determine the path which packets traversed through the network (Sager, 1998; Stone *et al.*, 2000). The upside of this approach is traceability of an attack long after it has completed. Obviously, the downside it is not scalable. Snoeren *et al.* (2002) propose to mark within the router to reduce the size of packet log and to provide confidentiality using a hash-based logging method. Stefan *et al.* (2001) suggest probabilistically marking packets as they traverse through routers. They propose router marking the packet with either the routers IP address or the edges of the path that the packet traversed to reach the router. With router based approaches, the router is charged with maintaining information regarding packets that pass through it. Most of above approaches are focus on DDoS attacks. Since our interest is not on events related to quick attacks, the work presented in this chapter differs from all above works.

4.3 Methodology

As mentioned in Chapter 3, the environment we are working is noisy due to the motivation uncertainty. By just looking at an event it is impossible to simply judge its motivation. As Davidoff and Ham claim other information such as contextual information (i.e. using a multivariate approach) can be useful to narrow down the meaning of such an event (Davidoff & Ham, 2012). For example a suspicious port scanning activity may have following characteristics: a single source address, one or more destination addresses, and target port numbers increasing incrementally. When fingerprinting such traffic multiple elements are examined to develop a hypothesis for the cause of behaviour (Davidoff & Ham, 2012).

By following the above idea, a multivariate approach is proposed for our tracing algorithm which profiling plays a key role. The formula 3.4 in section 3.3.2.2

is used for profiling assuming that immediate predecessor (or successor) of any node can be certainly found. In fact immediate hop can always be found by looking at the physical path of packet arrivals. Accumulated probability term (i.e. $\sum_t p(H_k/E)$, t is time) is divided by number of nodes behind the target hop (i.e. n_k - size of the subnet behind hop k if there is one) and known as *channel profile* score. The channel profile score is used as a measurement of the level of suspicion of channel k. The underlying idea is that by aggregating information of large volume of events it is possible to build a clear set of benchmarks of what should be considered as normal over extended period of time and hence to identify channels that anomalous data comes to the victim node. If the topological information is available let:

$$c_{kt} = \frac{\sum_t p(H_k/E)}{n_k} \tag{4.1}$$

is the *channel profile* score of k^{th} channel at time t. Then

$$z_{kt} = \frac{c_{kt} - \bar{c_t}}{\sigma_t} \tag{4.2}$$

is the Z-score of channel k at time t. where $\bar{c_t} = \frac{\sum_i c_{it}}{n}$, $\sigma_t = \sqrt{\frac{\sum_i (c_{it} - \bar{c_t})^2}{n-1}}$, and $i = 1, 2, 3, ..., n$.

4.3.1 Tracing algorithm

This section presents our algorithm for tracing slow suspicious nodes. The tracing algorithm has two segments: *tree formation* and *tree traversal*. Tree formation builds an equivalent tree structure for a given attack scenario enabling tree traversal to move towards the attacker's physical source. Algorithms for both process are given below.

4.3.1.1 Tree formation

Tree formation is the process of building an equivalent tree structure for a given attack scenario. Victim node is the starting point. Gateway node to victim is

considered as root of the tree and all immediate visible nodes (either internal or external) to the root are considered as children of the root. If a given child is a host node in the network then it becomes a leaf of the tree. If it is a gateway then it becomes a parent node of the tree and all immediate visible nodes to that node are attached as its children. This process is continued until the entire topology is covered (see Figure 4.2).

ϑ: Tree;
ω: A node;
τ: Set of all nodes in the given topology;
input : Topological information together with victim's location
output: Tree structure for the given attack scenario
Initialize the tree ϑ to have the root as the gateway of the victim;
List all nodes into the list τ;
/* attached each node to the tree*/;
tree-construction(ϑ,τ);
foreach *node* ω *in* τ **do**
    if *num-of-hops-between*(ϑ,ω)==*1* **then**
        insert ω into ϑ;
    end
end
foreach ϑ.*child* **do**
    tree-construction(ϑ.child,τ)
end

Algorithm 3: Tree formation for a given attack scenario.

4.3.1.2 Tree traversal

Once the equivalent tree structure is built, proposed tree traversal algorithm is applied. To traverse a non-empty tree, perform the following operations recursively at each node, starting from the root of the tree, until suspected node is found.

1. Visit the parent node

2. Compute channel scores for all children of the parent

3. Traverse the highest channel scored sub tree (if an attacker node is found backtrack to the parent)

4. Traverse next highest channel scored sub trees (only sub trees significantly deviate from rest of nodes of same parent)

The algorithm continues working towards a built tree node by node, narrow downing the attack source to one network and then to a node. At this point we can run more standard trace back methods by contacting the entity which controls that network if it is beyond our control.

input : A Tree constructed for anonymous slow attack scenario
output: A node where attacker is located
proposed-traverse(ϑ);
while *not found* **do**
    visit node ω;
    if *node* ω *is a leaf* **then**
        return;
    else
        profile all children of node;
        proposed-traverse(node.top_scored_child);
        proposed-traverse(node.next_scored_child);
    end
end

Algorithm 4: Tree traversal for a given tree.

4.4 Experiment

Figure 4.1 presents the network topology used for our experiments. Subnet sizes are as follows: S1(25), S2(25), S3(50), S4(25), S5(15), S6(25), S7(25), S8(5), S9(5) and Server farm(10). Following scenario is defined to test our approach.

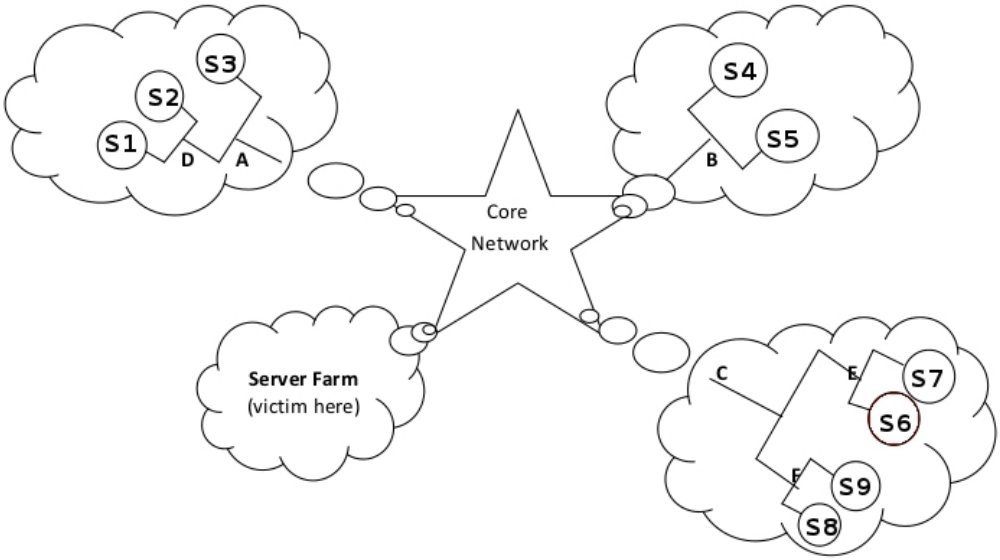

Figure 4.1: Network topology used for the experiment.

4.4.1 Scenario

Suppose that security staff have noticed someone's suspicious activity on a node in the server farm (see Figure 4.1) for sometime. Though they have a packet capture of the activity, they can't figure out what's going on, whether those events are a result of simple user mistakes or a malicious attempt. Origin of packets seems to be fake and the time gap between two consecutive events of that particular activity seems to be significantly high.

Above scenario is simulated using *NS3* by considering two cases: single and multiple attackers. In single attacker case an attacker is located at a node in subnet *S6*, and in multiple attackers case three attackers are located one in each in three different subnets *S3*, *S5* and *S6*. Network traffic is generated as described in section 3.4.1.2 ensuring that the source anonymity for attack events. Prior probabilities and Likelihoods are assigned as described in section 3.4.1.3. Equally likely assumption, i.e. $p(H_1) = 0.5$ in Equation 3.6, is followed for the single attacker case. But for multiple attacker case a slightly higher chance (55%) of sitting back the attacker at a node outside to the server farm is assumed and initial prior probabilities are assigned accordingly (i.e. $p(H_1) = 0.55$ in Equation 3.6). Each simulation was run for a reasonable time period to ensure that enough traffic

was generated (over one million events).

4.5 Results

Figure 4.2 presents the equivalent tree structure produced by Algorithm 3 for above scenario. The *root* denotes victim node while g_{i_j} and h_{i_j} denote a gateway or a host node at level i in Figure 4.2. j is a node number. Dashed rectangles represent a collection of leaves corresponded to hosts in each subnet. Once the tree is obtained, Algorithm 4 is run to locate the attackers as shown in Figures 4.3 to 4.6 for single attacker, and Figures 4.7 to 4.14 for multiple attackers.

Figure 4.3 represents the Step 1 of tracing process created at the root of the derived tree. Min and Max represent the minimum and maximum Z-scores of all immediate visible nodes (11 in total, except g_{1_3}) to the root at each time point. Since that graph suggests moving towards g_{1_3}, Step 2 graph is created at node g_{1_3}, and so on. Finally search is narrow downing to the subnet *S6*. Step 4 graph (see Figure 4.6) is created at *S6*'s gateway node g_{3_4}. In that graph A denotes the Z-scores corresponded to the true attacker located in that subnet. Min and Max represent the minimum and maximum Z-scores of all other nodes in subnet *S6*. Note that T denotes the threshold (Grubbs' critical value) which is not defined when number of data points in a set is two or less than two. In that case the highest scored path is chosen to move towards (see Figure 4.4) in finding attacker or directions to her location.

A similar manner should be followed in interpreting graphs in Figures 4.7 to 4.14 obtained for multiple attackers. In that case, once an attacker is found tracing algorithm should be back tracked to its immediate parent node and should proceed with next highest Z-scored sub tree if it deviates significantly from majority to find other suspicious nodes. Step 1 at Figure 4.7 depicts such a situation. After Steps 3 and 7, algorithm back tracks to the root node. Table 4.1 summarises travel sequences for tracing single and multiple attackers.

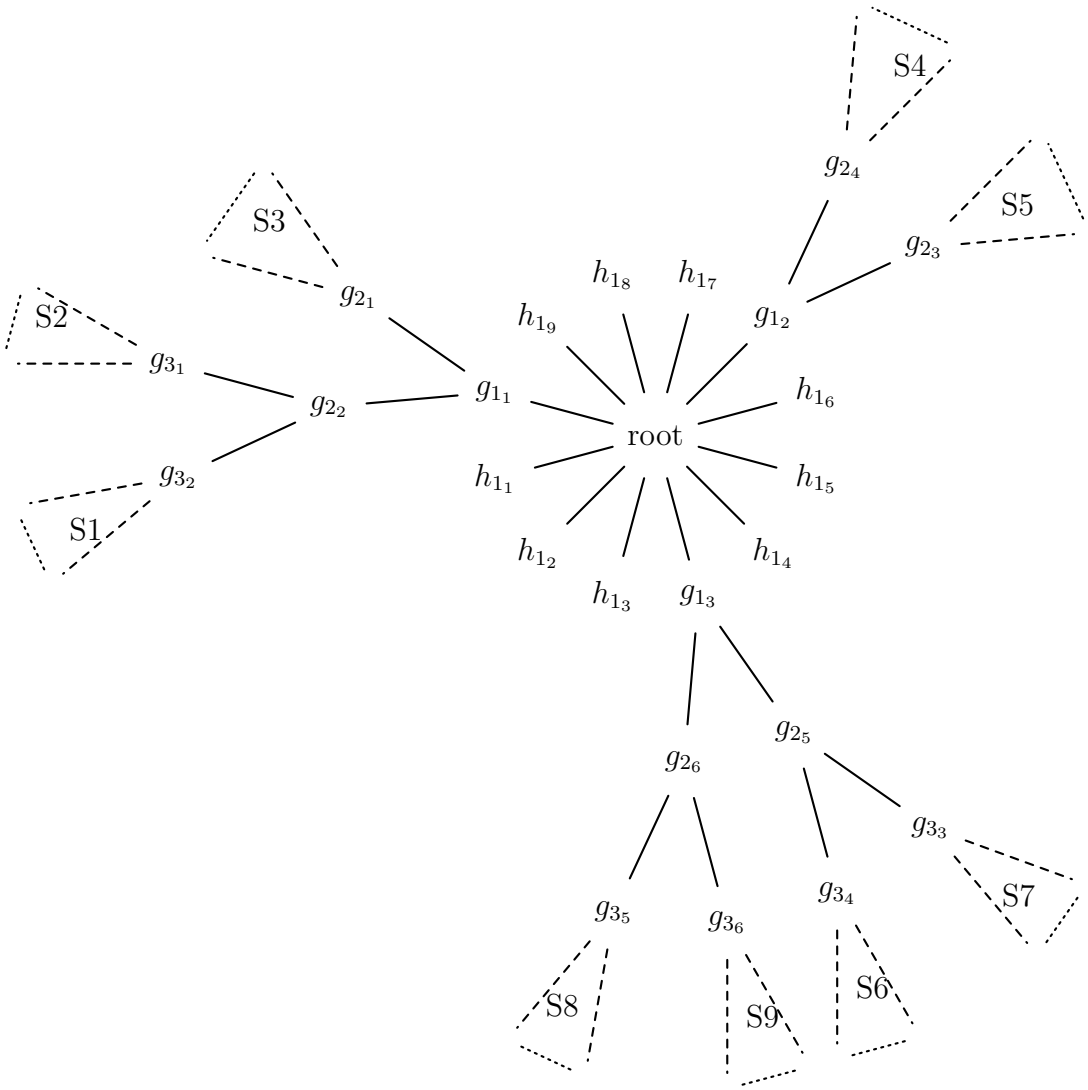

Figure 4.2: Equivalent tree structure for the given scenario.

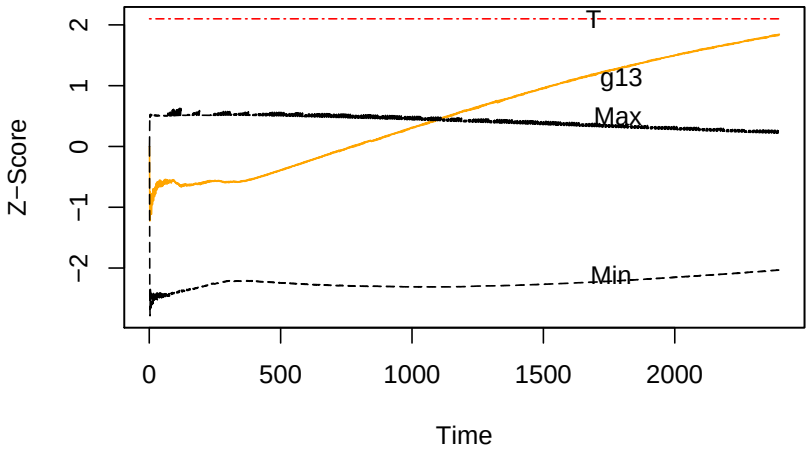

Figure 4.3: Single attacker Step 1 - node scores at *root*.

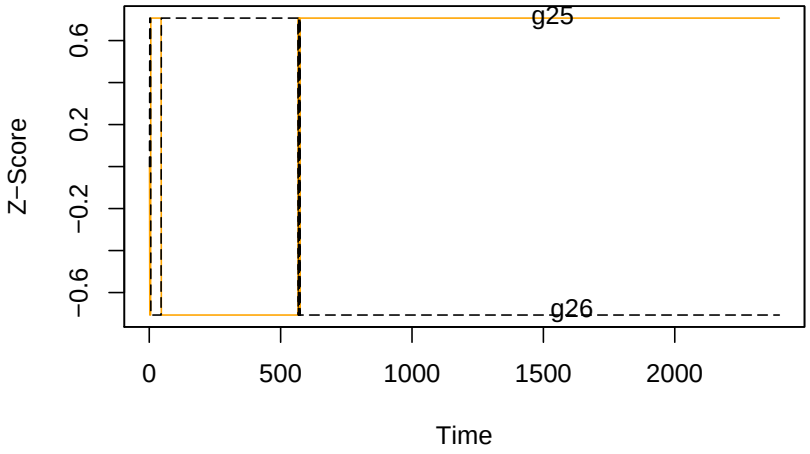

Figure 4.4: Single attacker Step 2 - node scores at *g13*.

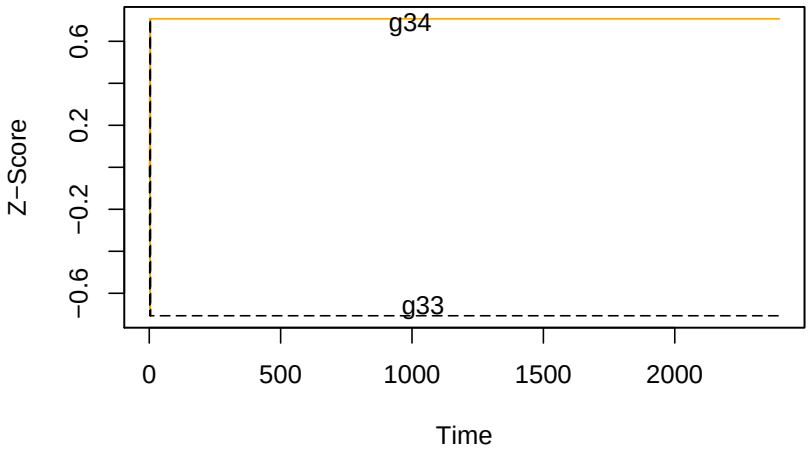

Figure 4.5: Single attacker Step 3 - node scores at *g25*.

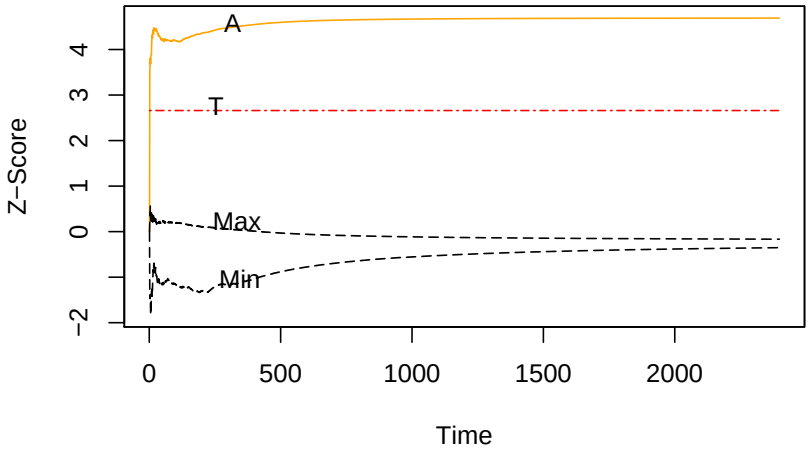

Figure 4.6: Single attacker Step 4 - node scores at *g34*.

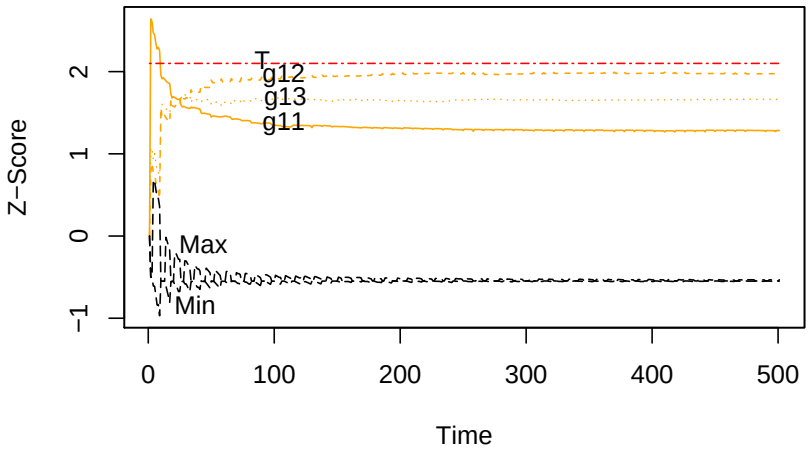

Figure 4.7: Multiple attackers Step 1 - node scores at *root*.

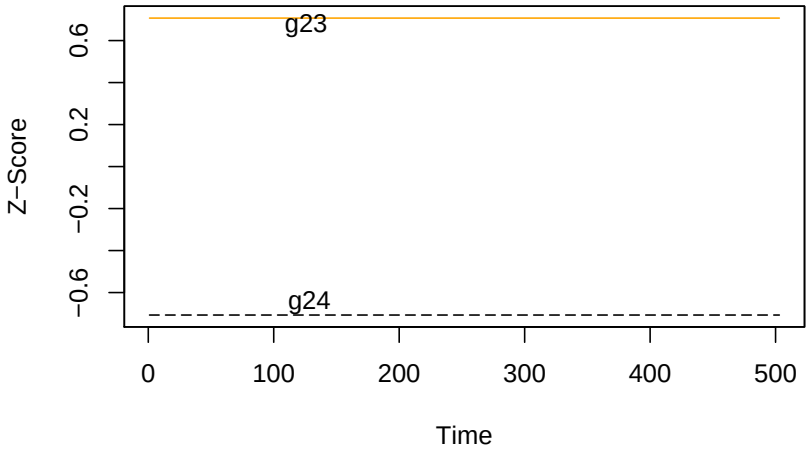

Figure 4.8: Multiple attackers Step 2 - node scores at *g12*.

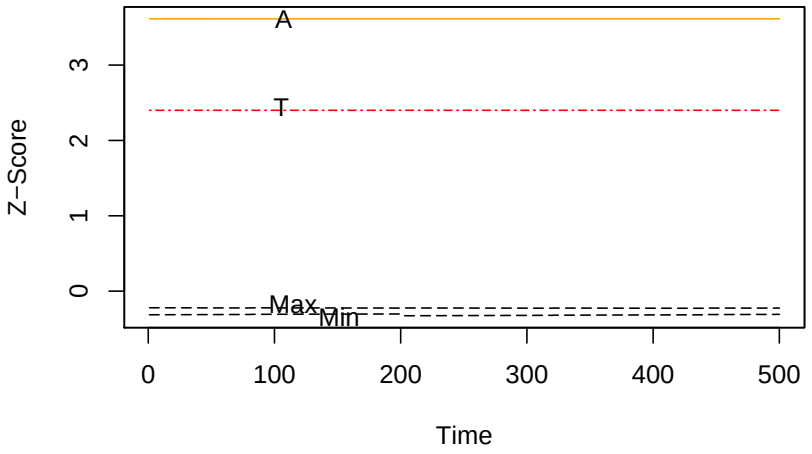

Figure 4.9: Multiple attackers Step 3 - node scores at *g23*.

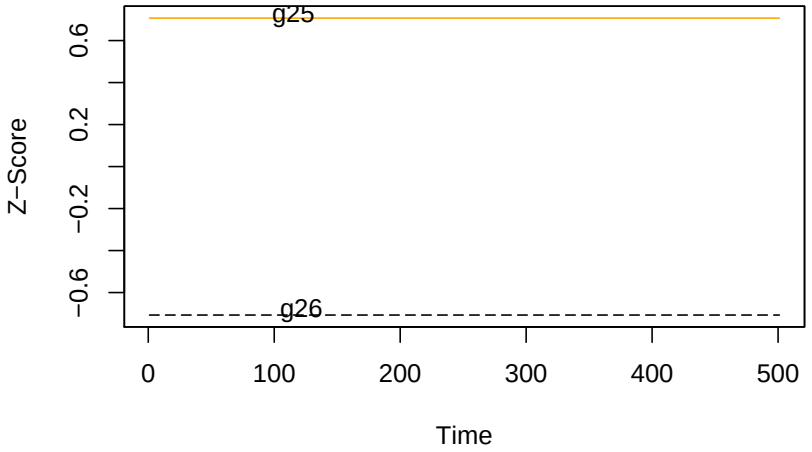

Figure 4.10: Multiple attackers Step 4 - node scores at *g13*.

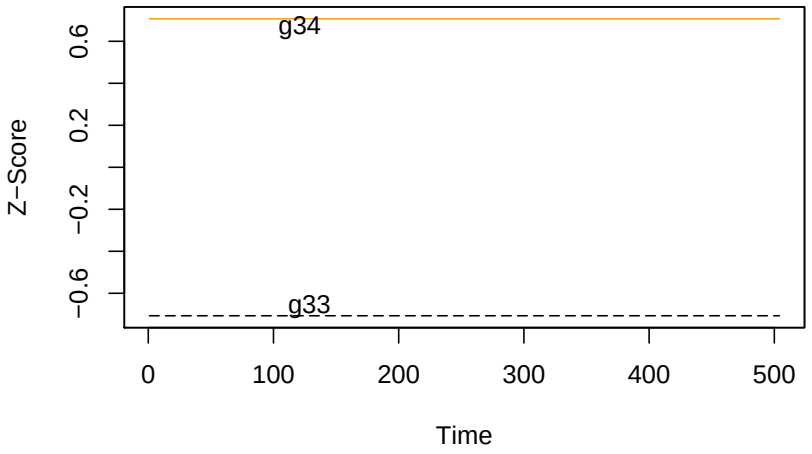

Figure 4.11: Multiple attackers Step 5 - node scores at *g25*.

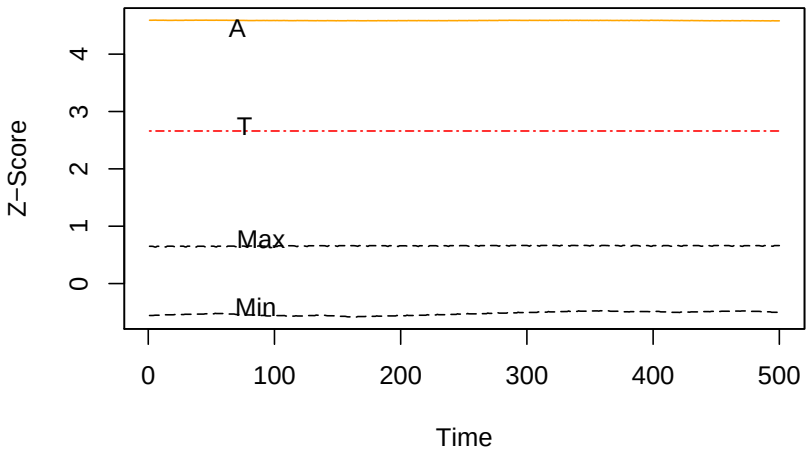

Figure 4.12: Multiple attackers Step 6 - node scores at *g34*.

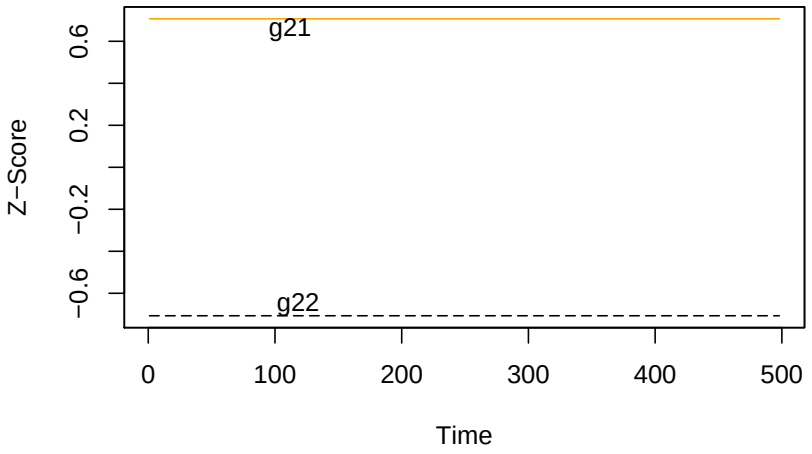

Figure 4.13: Multiple attackers Step 7 - node scores at *g11*.

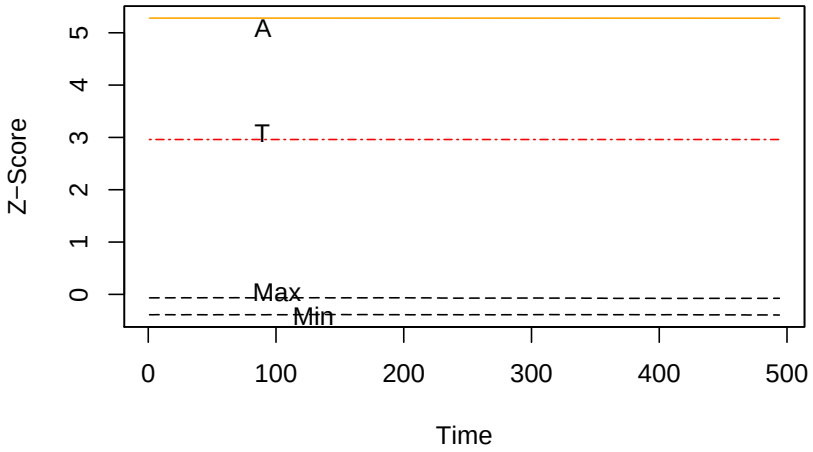

Figure 4.14: Multiple attackers Step 8 - node scores at *g21*.

Scenario	Travel sequence (until all attackers are found)
Single attacker	root, g_{1_3}, g_{2_5}, g_{3_4}
Multiple attackers	root, g_{1_2}, g_{2_3}, root, g_{1_3}, g_{2_5}, g_{3_4},root, g_{1_1}, g_{2_1}

Table 4.1: Traversal sequences for tracing attackers.

4.6 Discussion

Requiring topological information of monitoring network is a limitation of our approach. This limitation occurred due to denomination factor (n_k) in Equation 4.1. However automated tools and techniques are available at present for obtaining topological information (network layer map) of a given network (Tozal & Sarac, 2012), and such a tool can be integrated with our algorithm to overcome that limitation. These tools are capable of capturing a network layer map with various topological characteristics such as total number of routers, degree distribution of routers, average router degree and betweenness (Tozal & Sarac, 2012). Introducing a normalization factor for n_k is another option to overcome that limitation and hence to extend our approach for any network.

Since our focus is slow activities we assume that there is enough time to move equipment and programs into place, map routes, and perform the actual trace back. Our approach assumes that attacker is not moving to other networks as a part of launching the attack. Therefore tracing collusion activities still remain unsolved. We acknowledge that use of sophisticated attack activities such as use of bot-nets, throwaway systems and distributed sources makes it very difficult to tracing down slow activities. Of further interest is to determine the target of such activity as a part of defensive mechanism (see Chapter 6).

It is possible to automate the proposed approach for real world implementation by developing simple scripts which enables remote data collections from network devices (Braun *et al.*, 2010). For example Satten (2007) shares a collection of code, methods and insight for capturing packets remotely from a network without connecting a capture box to the remote network. Our approach can be implemented on that type of code base to improve the performance of remote capturing.

There is a possibility to extend this work as an adaptive approach to network traffic analysis would also be welcome to address selective monitoring and collection of packets. Little research has considered this problem. One hardware-based approach to characterise unlikely uninteresting traffic more cheaply that can be devoid of further more expensive software-based analysis exists (Gonzalez *et al.*, 2007); this is demonstrated to be effective for potential Gigabit Ethernet operations. However, further work is needed to allow for traffic monitoring to be sensitive to the type of services a given node may be vulnerable against. This will help avoid undue attention to suspicious traffic that will not prove harmful.

4.7 Conclusion

In a situation there are multiple suspected sites to be investigated (eg, different actors, subnets, LANs, locations etc) prioritization centres of attention would be a problematic. Localisation attackers' identities as much as possible, at least for an intermediary level, or choosing the smallest subset which attacker may be included would greatly save the cost and time to be spent for investigations. Our approach serves on this task.

This approach is independent from the subnet size. Increasing or decreasing subnets' size, but keeping the same topological structure, does not change the number of steps required in tracing process. Our approach changes its traversal sequence according to the suspicious traffic. As a result, suspicious node comes forward in the sequence and probability of trapping it early becomes high.

Requiring topological information is a limitation of our approach at present and hence specially suitable for tracing down anonymous insiders. In future we hope to further develop this approach for any network. Finally the main technical finding of this chapter is the tracing algorithm.

Chapter 5

Lightweight monitoring

This chapter explores possibility to minimise data collection for our monitoring algorithm by employing traffic sampling, and examines effects of network design on sampling error. The main contribution of this chapter is proposing a lightweight monitoring scheme.

5.1 Introduction

Network monitoring for traffic analysis ranges from counting the volume of traffic to capturing packets transferred throughout the network using various methods (Shaikh *et al.*, 2009). As contemporary enterprise networks scale up in size and speed, huge volume of traffic has a cost ramification for collection infrastructures. Resources of network devices are comparatively expensive and scarce. Such resources need to be utilised on their regular activities than utilising on monitoring activities. As volume and rate of traffic are rising, inspection of each and every individual packet is not feasible any more. A data reduction is needed and could be motivated as long as it preserves the required level of precisions for monitoring objectives.

Employing statistical sampling and estimating interested *security parameters* of entire population by analysing a small unbiased sample would be a possible method for data reduction. If a drawn sample faithfully represents entire traffic characteristics of interested parameter then it can be used to analyse. It helps

with data reduction in terms of both traffic collection and processing. Traffic sampling approaches have been suggested by the Internet Engineering Task Force (IETF) working groups (IETF, 2009), and already been employed by some tools (e.g. Cisco NetFlow (Cisco, 2013)) in their router design. Note that random sampling techniques have a distinct advantage against other alternative methods for data reduction. It allows retention of arbitrary details while other methods for data reduction (e.g. filtering and aggregation) require knowing the traffic features of interest in advance.

Despite all the benefits mentioned above, there is an inherent tension and debate of using traffic sampling for security specific tasks. Obviously, signature based detection methods can be seriously affected by sampling as selection of only a subset of signature elements would not be sufficient to recognise a predefined signature pattern. But in anomaly based detections, should whole traffic still need to be investigated? By definition an anomaly of a parameter of interest is a deviation of a computed statistic of that parameter from a norm of the normal traffic statistics. If sampling changes the statistics of normal and anomalous traffic equally, it is reasonable to hypothesise that detectability would not be affected by the sampling rate.

This chapter presents our study on impact of traffic sampling on our monitoring algorithm presented in Chapter 3. The study has two objectives: 1- investigating the feasibility of employing traffic sampling with our monitoring algorithm to produce a more light-weighted version of the algorithm, and 2- examining how design of the network affects on sampling error.

5.2 Related work

Objectives of network monitoring can be classified into three main categories: traffic engineering, accounting and security specific. The accuracy requirements of these applications are quite different. Traffic sampling for security specific tasks (particularly anomaly detections) has not been comprehensively studied when it compared with number of studies available on traffic sampling for engineering and accounting tasks. Using sampling for traffic engineering and accounting tasks is widely studied (Duffield, 2004) and already been employed by commercially

available tools (Cisco, 2013). However those studies are not relevant to this work as our objective is a security specific. Because a sampling technique which performs well in traffic engineering and billing tasks might not be necessarily a good choice for effective anomaly detection (Jurga & Hulb, 2007).

Using sampling for security specific objectives can be found in (Ali *et al.*, 2010; Ishibashi *et al.*, 2007; Claffy *et al.*, 1993; Tellenbach *et al.*, 2008; Mai *et al.*, 2006a,b; Barakat *et al.*, 2005; Brauckhoff *et al.*, 2006; Fazio *et al.*, 2012; Taylor & Alves-Foss, 2001; Reeves & Panchen, 2002). But none of them focuses on slow activities. Note that methods proposed for typical rapid attacks cannot be used to monitor for slow activities due to several constraints including limitations of computational resources (Chivers *et al.*, 2013). To the best of our knowledge this is the first attempt to use sampling technique for slow activity monitoring in computer networks.

Based on sampling frame, existing sampling proposals can be classified into two groups: packet-based and flow-based. Packet-based techniques (Ali *et al.*, 2010; Ishibashi *et al.*, 2007; Claffy *et al.*, 1993; Tellenbach *et al.*, 2008; Mai *et al.*, 2006a,b; Duffield *et al.*, 2005; Reeves & Panchen, 2002) consider network packets while flow-based techniques (Hohn & Veitch, 2006; Duffield *et al.*, 2002; Mai *et al.*, 2006a; Androulidakis & Papavassiliou, 2008; Androulidakis *et al.*, 2009; Bartos & Rehak, 2012) consider network flows as elements for sampling. Packet sampling is easy to implement as it does not involve any processing before selection of samples. But in the case of flow sampling, monitored traffic is processed into flows first and then apply sampling technique on whole set of flows for drawing a sample. This requires to use more memory and CPU power of network devices. Yang & Michailidis (2007) is a study of combination of packet and flow sampling. A comparison of packet vs flow sampling can be found in (Hohn & Veitch, 2006). The most widely deployed sampling method in literature is packet sampling. It is computationally efficient, requiring minimal state and counters (Tellenbach *et al.*, 2008).

Ali *et al.* (2010) propose an algorithm to sample malicious packets with higher rates to improve the quality of anomaly detection. High malicious sampling rates are achieved by deploying in-line anomaly detection system which encodes a binary score (malicious or benign) to sampled packets. Packets marked as malicious

are sampled with a higher probability. Obviously this approach involves additional processing and storage overheads. Ishibashi *et al.* (2007) evaluate quantitatively how sampling decreases the detectability of anomalous traffic. They show that by changing the measurement granularity it is possible to detect anomalies with low sampling rates and to use relationship between the mean and variance of aggregated flows to derive optimal granularities.

Claffy *et al.* (1993) investigate the performance of various methods of sampling in network traffic characterization. They use several statistics to compare two distributions for similarities in order to compare sample traces with their parent population. Tellenbach *et al.* (2008) evaluate effect of traffic mix on anomaly visibility using traces collected at four different border routers. They use prior knowledge of two different worm types to measure the visibility level of anomaly at various sampling rates. Mai *et al.* (2006a,b) are evidence to show that suitability of a sampling technique depends on the detection method. Former investigates how packet sampling impacts on three specific port scan detection methods. The same work has been extended in later to investigate impact of other sampling methods. Both studies conclude that packet sampling is less effective on anomaly detection when using their detection method. Brauckhoff *et al.* (2006) show that entropy metrics are less affected by sampling and are able to expose the blaster worm even at high sampling rates.

Event based and Timer based are two possible mechanisms to trigger the selection of a sampling unit for inclusion in a sample. Event based approaches collect one elements out of N elements using chosen sampling method. Naive 1 in N sampling strategy by Cisco NetFlow (Cisco, 2013) is a well known example for that method. It samples one packet after every N packets. Event based approaches consume more CPU and memory of network devices as it involves some processing (counting). In a timer based approach one packet is sampled during N time units. This approach is effective in terms of CPU and memory consumptions as it depends on system timer. However choosing larger Ns returns higher sampling errors due to non-time-homogeneous nature of packets arrivals to the network.

5.3 Lightweight monitoring

As mentioned above decision to inspect each and every packet (or flow) can be expensive for collection infrastructures in terms of time, bandwidth and computational resources, and also may not feasible in modern day Gigabit networks (Jurga & Hulb, 2007). This section proposes a *Lightweight monitoring* scheme for slow activities. It minimises amount of information needed to collect from different parts of the network while still be able to distinguish slow suspicious activities from normal activities. First traffic is sampled and then monitoring algorithm is applied on drawn samples.

5.3.1 Sampling methodology

Network data constitutes a potentially unlimited population continuously growing up by the time. Hence whole observation window W is segmented to number of smaller time windows w and traffic is sampled within smaller windows using *stratification* sampling technique with *optimum allocation* method, which is designed to provide the most precision for the least cost (Trek, n.d; Dıaz-Garcıa & Cortez, 2006). If h is a traffic stratum the best *sample size* n_h for stratum h is given by:

$$n_h = n.\frac{\left[\frac{N_h.s_h}{\sqrt{c_h}}\right]}{\sum \frac{N_i.s_i}{\sqrt{c_i}}} \tag{5.1}$$

where n_h-sample size for stratum h, n-total sample size, N_i-population size for stratum i, s_i-standard deviation of stratum i, and c_i-direct cost (in terms of time, bandwidth, and computational resources) on the collection infrastructure to sample an individual element from stratum i. Note that the direct cost should be in a common unit (CU) of measurement for the amount of computational cost spending on different parameters. The time, bandwidth, memory or processor requirements that constitutes one common unit (1CU) varies based on which requirement is being measured, and how each parameter is critical and scarce to the network. Hence definition of such a unit (CU) would be subjective. For instance one can define: 1CU is memory equivalent of 128MB, 1CU is bandwidth equiv-

alent of 56KBPS, 1CU is CPU-Time equivalent of 100 nsec etc. International unit (IU) in pharmacology is a well-known example for a similar approach for a common unit of measurement for the amount of a substance (Ansel & Prince, 2004). The main advantage of above sampling technique is producing the most representative sample of a population to the least cost. Hence it is a reasonable choice to employ with our problem as "cost" parameter can be minimised, subject to the required precision, to obtain a light-weighted monitoring scheme. Traffic classification is employed to establish the strata. Using a basic classification technique (e.g. using known L4/L3 access lists and Protocols) would be enough as advanced classification techniques may consume more computational resources. All other traffic which is not belonged to a predefined stratum is pooled into a common stratum. A detailed discussion of possible types of network traffic is beyond the scope of this thesis. Note that running an advanced traffic classification algorithm to establish the strata is not recommended at this point as it can add processing overheads to collection infrastructure.

The simple random sampling (SRS) technique is used to select a n_h size sample from stratum h. Stratification ensures that each traffic type is adequately represented (Trek, n.d). Ensuring (as much as possible) heterogeneity among strata with homogeneity within stratum will increase the precision of sampling.

Each element of the population having a non-zero probability of selection is a preliminary condition for any random sampling technique. Sampling traffic simply from backbones or edge routers (as most existing works do[1]) seriously violates this condition in terms of security specific view. Because it ignores consideration of traffic within same broadcast domains, and hence potential insider activities. Therefore traffic is sampled at each broadcast domain in this work but considering incoming traffic only. Considering incoming traffic avoids selection of a given unit (packet or flow) twice for inclusion in a sample at source and destination points.

[1]Sampling traffic from backbones or edge routers is enough for most of accounting and traffic engineering tasks.

5.3.2 Monitoring algorithm

The algorithm presented in Chapter 3 is applied on drawn samples. It further reduces data sampled during w into a single value which is important to maintain information about node activities for a very larger time window W. However considering complex methods likes Principal Components Analysis (Smith, 2002) (combines variables based on the correlation) as the data fusion technique is purposely ignored as they introduce extra computational overheads to monitoring process. The probability term $p(H_1/E)$ in Equation 3.4 is accumulated by time to generate node scores for larger observation window W. Peer analysis technique described in section 3.3.3.3 is used at each time point to distinguish slow suspicious activities from normal activities.

5.3.3 A Case study

An attacker is located at a node in a subnet of network B (see Figure 3.1), and network traffic including suspicious activities is generated as described in section 3.4.1.2. Each simulation is run for a reasonable period of time to ensure that enough traffic is generated (over one million events). A simple R (R Development Core Team, 2010) script (see Appendix B.4) is written to sample packets as described above. Prior probabilities and Likelihoods are assigned as in section 3.4.1.3. c_i in Equation 5.1 is set to a constant value assuming there is no significant difference between stratum for the cost of selecting an element for inclusion in a sample.

5.3.3.1 Attacker detection

A series of experiments have been conducted by changing the sampling rate r, hence n in Equation 5.1. Figure 5.1 presents the outcome of the proposed approach when $r = 10\%$ of the whole traffic N. *Min* and *Max* represent the minimum and the maximum profile scores of normal nodes in the subnet where attacker node A is located. T represents the Grubbs' critical value (threshold) for attackers' subnet. As it is obvious from Figure 5.1, our algorithm together with chosen sampling technique is capable of detecting slow activity using a 10% size sample.

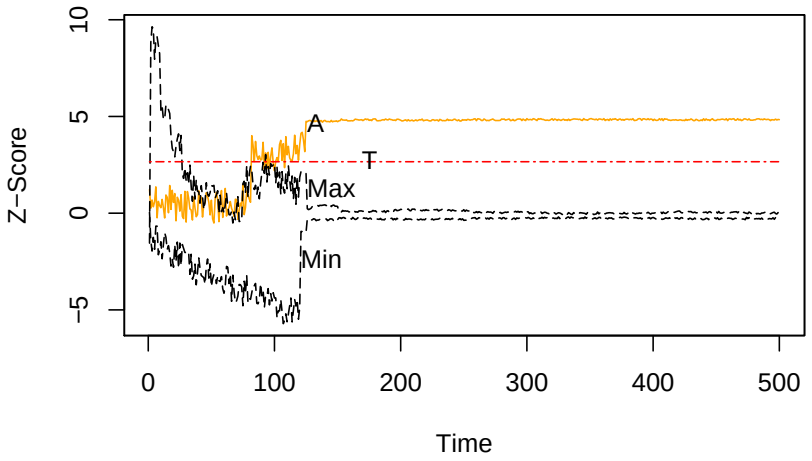

Figure 5.1: Running the detection algorithm over 10% size sample.

5.3.3.2 Detection potential

Detection potential measures how likely an activity could be detected as a suspicious slow activity. It can be expressed in terms of deviations of profile scores from the threshold line. Obviously, the higher the deviation (order the real values of deviations, not absolute values) has the better chance of detection. On that basis the detection potential D is defined as: $D = z - T$. Figure 5.2 compares the *detection potential* against the sampling rate r. It is obvious that a point of diminishing returns is existed in Figure 5.2. When r is *larger enough* to produce a reasonable level of accuracy making it further large simply wastes resources of monitoring infrastructure.

5.4 Network design

A sampling process has two types of errors: *sampling* and *non-sampling*. Sampling error occurs because of the chance, and it is impossible to avoid but can be minimised by defining unbiased estimators with small variances. Non-sampling errors can be eliminated, and occurred due to many reasons: inability to access

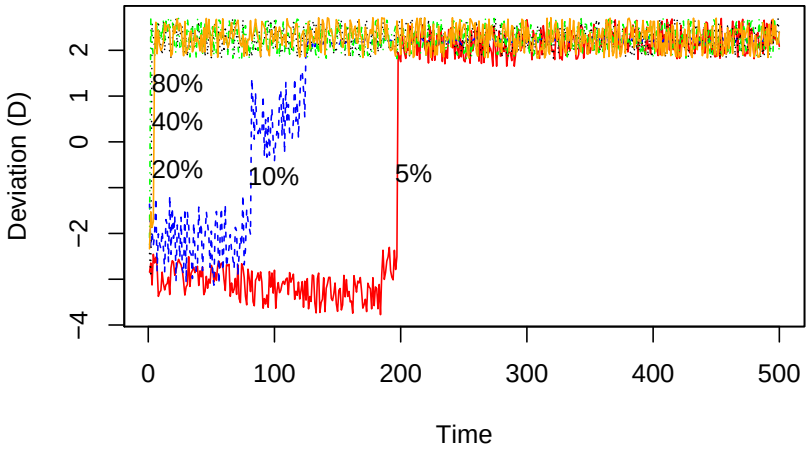

Figure 5.2: Detection potential vs sampling rate.

information, errors made in data processing, etc (Tozal & Sarac, 2012). This section examines what impact would varying network size and subnet structure have on *sampling error*.

Proportion of anomaly packets ϕ is considered as the parameter of interest for this analysis and hence sample proportion π is defined as $\pi = (a/n)$; where a is the number of suspicious packet in a given sample size n. Note that proportion of illegitimate to legitimate traffic, i.e. $a : (n - a)$, is a dominating factor for likelihood of false alarms in an IDS (van Riel & Irwin, 2006b). Though the distribution of ϕ is binomial, in a network scenario, this can be approximated by a normal distribution given a overwhelm number of packets to deal with (it satisfies the conditions of $n.\hat{\pi} \geq 15$ and $n.(1 - \hat{\pi}) \geq 15$). Hence, $\phi \sim Normal\left(\hat{\pi}, \sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n}}\right)$, where $\hat{\pi}$ is the observed proportion from samples. This can be used to draw inference about the unknown population proportion ϕ.

5.4.1 A Case study

An attacker is located in a 224 size network and $\hat{\pi}$ is estimated in each case as described below. Following the Monte Carlo technique to simulations (Sawilowsky,

Sampling rate(r)	5%	10%	20%	40%	80%	Whole trace
$\hat{\pi}$	0.00038	0.00034	0.00036	0.00035	0.00036	0.00036
P.Value	0.0970	0.0929	0.0952	0.0971	0.9770	N/A

Table 5.1: Proportion over sampling rates.

2003), each simulation was repeated over 100 times. Goodness-of-fit test (Rao & Scott, 1981) is applied to statistically test the independence (or homogeneity) of proportion π over *sampling rates*, *number of subnets* and *subnet sizes*. If any dependency is found it is depicted in a graph as shown in Figures 5.3 and 5.4.

5.4.1.1 Sampling rate (r)

Traffic samples at 5%, 10%, 20%, 40%, and 80% rates of the whole trace were drawn and $\hat{\pi}$ was calculated. The null hypothesis H_0 is the assertion that the sample proportion π conforms to the whole traffic proportion ϕ (i.e. H_0:$\forall r\ \pi_r$=ϕ). The alternative hypothesis H_1 is the opposite of H_0 (i.e. H_1:$\exists r\ \pi_r \neq \phi$). $\hat{\pi}$s and p-values of testing H_0 vs H_1 are given in Table 5.1 where p-values are greater than the significance level $\alpha = 0.01$ for all cases. Therefore there is no enough evidence to reject the null hypothesis H_0. Hence we conclude that sample proportion π conforms to the whole traffic proportion ϕ. In other words π can be used to draw inference about ϕ.

5.4.1.2 Number of subnets (b)

An attacker is located in a 224 size network and same experiment was repeated for four more times by changing the number of subnets, which was doubled at each time, but keeping all other conditions unchanged. The null hypothesis H_0 - the assertion that the proportion π is not affected by the number of subnets b (i.e. H_0: $\pi_0 = \pi_2 = \pi_4 = \pi_8 = \pi_{16} = k$) vs alternative hypothesis H_1 - the opposite of H_0 (i.e. H_1: $\exists b\ \pi_b \neq k$) was tested. $\hat{\pi}$s and p-values of testing H_0 vs H_1 are given in Table 5.2. Since p-values are less than the significance level $\alpha = 0.01$ for some cases we conclude that there is no enough evidence to accept the null hypothesis: "proportion π is not affected by the number of subnets b", which means that proportion is affected by the number of subnets. Figure 5.3

Number of Subnets(b)	0	2	4	8	16
$\hat{\pi}$	3.58E-04	2.86E-03	1.12E-04	8.52E-05	1.97E-05
P.Value	N/A	2.65E-01	6.03E-06	3.94E-07	1.04E-11

Table 5.2: Proportion over Number of Subnets.

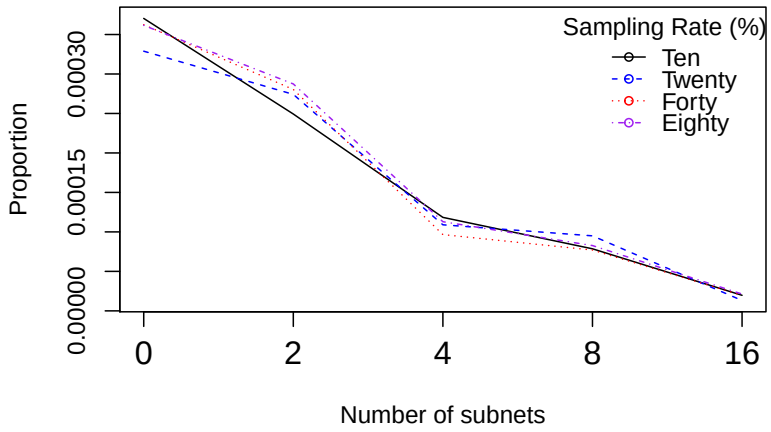

Figure 5.3: Proportion vs Number of subnets at each sampling rate.

presents the relationship between number of subnets b and proportion π. When b is increased $\hat{\pi}$ is decreased (deviates from the actual value).

5.4.1.3 Subnet size (n)

An attacker was located in a 5 nodes size subnet in the network, and $\hat{\pi}$ was calculated at each sampling rate. The same experiment was repeated for different subnet sizes: 10, 20, 40, and 80 without changing other parameters. The null hypothesis H_0 - the assertion that the proportion π is not affected by the subnet size n (i.e. H_0: $\pi_5 = \pi_{10} = \pi_{20} = \pi_{40} = \pi_{80} = k$) vs alternative hypothesis H_1 - the opposite of H_0 (i.e. H_1: $\exists n\ \pi_n \neq k$) was tested. $\hat{\pi}$s and p-values of testing H_0 vs H_1 are given in Table 5.3. Since p-values are less than the significance level $\alpha = 0.01$ for some cases we conclude that there is not enough evidence to accept the null hypothesis: "proportion π is not affected by the subnet size". Figure 5.4

Subnet Size(n)	5	10	20	40	80
$\hat{\pi}$	7.28E-04	8.61E-04	8.84E-05	2.06E-04	5.24E-05
P.Value	2.20E-16	2.20E-16	2.80E-01	6.39E-04	N/A

Table 5.3: Proportion over Subnet sizes.

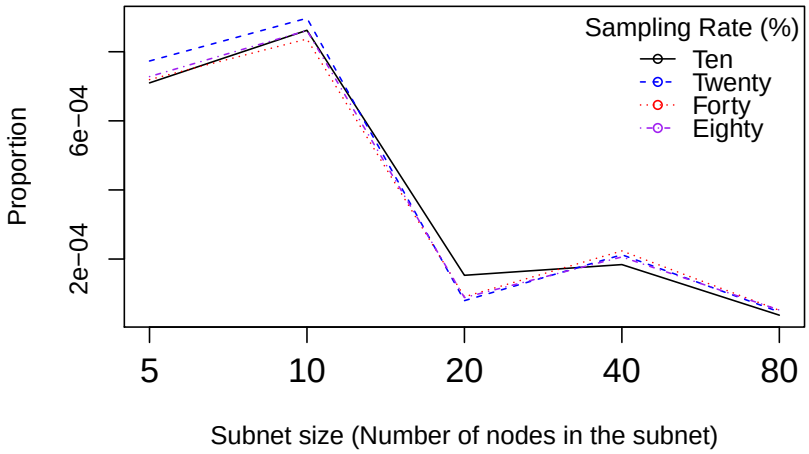

Figure 5.4: Proportion vs Subnet size at each sampling rate.

presents the relationship between subnet size n and proportion π: when n is increased $\hat{\pi}$ is decreased in overall (deviates from the actual value).

5.5 Discussion

Section 5.1 raised an important question "Do we need to look at whole traffic in slow activity monitoring?" As it is obvious from Figure 5.2, when $r \geq 20\%$ there is no significant difference in detection potential d. Though a point of diminishing returns is existed between 10% and 20% for this particular simulation scenario, authors do not claim it generally for any algorithm. Instead we argue that a point of diminishing returns can be existed for a given anomaly based detection technique, and finding it leads to save resources of monitoring infrastructure.

However, a very practical question occurred here is whether analysts are capable of finding the size of "larger enough" sample (point of diminishing returns) of their real world networks in day-to-day operations? A similar question has raised in (Ishibashi *et al.*, 2007). We leave this to pursue as future work.

Even for signature based methods, it should be noted that there are number of malicious activities that can be detected with 100% accuracy even using a single anomaly packet size sample. A protocol violation and a connection to an IP address from a blacklisted NAT device are examples for such activities (Jurga & Hulb, 2007). A single packet would also be sufficient for host and OS identification and such identification is often used for masquerade detection (Phaal, 2013; Zalewski, 2013). Obviously, there are network anomalies which cannot be detected by observing only a single packet (e.g. DNS tunnelling). For instance, a DNS tunnel could be detected by observing that the volume of data transferred over port 53 is much higher than usual. In order to improve the accuracy of detection, a minimum number of observations is required (Jedwab & Phaal, 1992). However, required level of sampling rate depends on several factors: parameter of interest, the statistic used for detection, detection algorithm, sampling method, level of precision required, duration of monitoring, rate of attack events etc. Further research is needed to identify group of security issues (parameters) that can sampling be utilised and to find relationship between above parameters.

As shown in section 5.4.1.1, sampling methodology used in this work draws representative samples for the proportion. Since the monitoring algorithm in this work does not match exact signature sequences for detections (instead it accumulates likelihoods), detectability is not considerably affected by the sampling rate of such samples (see Figure 5.2 and section 5.4.1.1). Interestingly, though detectability is not affected by the sampling rate it can be affected by the network design, particularly by number of subnets and subnet size (see sections 5.4.1.2,5.4.1.3). When either is increased, the estimated value is decreased and deviated from the original value regardless of data is sampled or not. Note that since we have used the proportion as the parameter of interest, in other way, this implies that how many suspicious events you may observe to make your decisions as a return to changing the network design. Studying the affects of network parameters such as network design and sampling rates on visibility level of anomalies is increasingly

popular among the research community. Tellenbach *et al.* (2008) is one such example where authors use the prior knowledge of two different worm types to measure the visibility level of the anomaly at various sampling rates.

It is important to highlight that the proposed approach is not intended to study "how to sample network traffic so that any anomalies or security issues can be identified by analysis of just the sampled packets and not requiring to inspect all packets in the network traffic". Our position sampling can not be employed to tackle "*any* anomalies or security issues" in general, instead it can be employed to tackle certain classes, groups of anomalies or security issues such as slow activities that can be difficult to detect using typical detection methods.

5.6 Conclusion

Decision to inspect each and every packets can consume more resources at network devices for packet processing, and more bandwidth for transmissions them to collection points. Since we propose to sample data at the place of each sensor it reduces the computational resources needed for packet processing and transmissions. Sampling together with information fusion technique can be employed as a data reduction method for slow activity monitoring. Sampling methodology used in this work effectively captures the required number of anomaly observations for the proposed monitoring algorithm even at smaller sampling rates. As shown in section 5.4 the level of detectability can be affected by the network design.

At this stage of the research, we have shown that there is a possibility to use sampling techniques in slow activity monitoring and some population characteristics remain unchanged in samples as well. It is important to understand what can and cannot be monitored using the proposed method and further research is needed. In general, there is a higher tendency by higher order statistics to show the same characteristics within samples as in the population. Therefore investigating the sampling behaviours of detection algorithms, which are based on higher order statistics, would be interesting. Lack of availability of such kind of algorithms anticipates for testing this idea at present.

Finally the technical findings of this chapter include:

- exploring feasibility of employing sampling technique with our monitoring algorithm to produce a more lightweight version;

- relationship between network parameters and sampling error.

Chapter 6

Target-centric monitoring

Unlike the source-centric approaches presented in previous chapters, this chapter presents a target-centric approach. We strive to neglect any kind of information about the source, and develop the method along the target alone to prove that shifting to target-centric approach brings major benefits with respect to other approaches. The main contribution of this chapter is proposing a shift to focus of analysis.

6.1 Introduction

As mentioned throughout the thesis a particular challenge is to monitor for slow and suspicious activities is deliberately designed to stay beneath detection thresholds. It becomes even more difficult if such activities are a result of collusion and/or anonymous. To detect any attack, monitoring systems must consider event stream as a function of time and should use signature or anomaly based methods against event flow. Most of existing monitoring schemes share a common feature which we called *source-centric* analysis. They perform analysis based on source information (in fact perceived last hop) of activity either it is a host based or a network based monitoring system, and utilise that information at some stage of detection assuming that suspicious activity can be attributed to a meaningful specific source or an intermediate (Whyte *et al.*, 2006). For example, as mentioned in (Peng *et al.*, 2004), most of the existing solutions become less effective when

the attack is launched from distributed sources. The actual reason behind such a deficiency is that their dependency on the Source IP addresses (or perceived last hop) of activities for attack detection, i.e. the source-centric analysis. What we propose in this chapter is we should move away from source-centric analysis to destination-centric analysis. Note that this should not be mistaken as a host based monitoring scheme, as it is not a problem of location of IDS deployment. It is a matter of whether monitoring system depends on source information of activities for attack detection or not.

As mentioned in section 3.3.1.2 there is no guarantee on publicly visible source of an event is to be true source as various methods let an attacker getting anonymous. Source centric monitoring schemes are vulnerable to these situations. Attacker tactics such as source collusion and source address spoofing are common and therefore make such attacker detection very hard. It becomes worst if the attack is a slow attack as current computational constraints of monitoring devices do not let to keep information about activities over extended periods of times to correlate between suspicious events.

To address this we propose a method that does not require correlating to a common source. We shift the focus away from potential sources of attacks to potential targets of such activity. The proposed approach is designed to utilise destination information of activities together with a data fusion technique to combine output of several information sources to a single profile score. We analyse for suspicious activities based on (or around of) the destination information only, but completely independent from the source information. This provides target node profiling where nodes suspected of being attacked are essentially monitored for and detected. Simply put, we shift the focus of analysis away from potential sources of attacks to potential targets of such activity. As a result resources devoted to detection and attribution could be redeployed to efficiently monitor for detection and prevention of attacks. The effort of detection should aim to determine whether a node is under attack, and if so, effectively prevent the attack.

6.2 Methodology

The problem of *target-centric* monitoring is also broken down into two sub problems: *profiling* and *analysis* as same as in source-centric monitoring approach in Chapter 3. Profiling provides for evidence fusion across spaces in a Bayesian framework and accumulation across time, and analysis distinguishes between anomalous and normal profiles using Grubbs' test. Note that we simply change definitions of hypothesis H_1 and H_2 defined in section 3.3.2.2 as follows to use them in the target-centric approach. If H_1 and H_2 be two possible states of a node in computer network, we define H_1 as *node under attack* and H_2 as *node not under attack*. Then H_1 and H_2 are mutually exclusive and exhaustive states. If E={e_1,e_2,e_3,...,e_m} is a set of evidences on proposition H_1 obtained through independent information sources then:

$$p(H_1/E) = \frac{\prod_{j=1}^{m} p(e_j/H_1).p(H_1)}{\sum_{i=1}^{2}\prod_{j=1}^{m} p(e_j/H_i).p(H_i)} \tag{6.1}$$

As described in section 3.3.2.3, the assumption of statistical independence of information sources is reasonable as we propose to use distinct types of information sources operated independently.

6.3 A Case Study

Network simulator $NS3$ is used to build a network topology consisting of a server farm and 10 subnets of varying size. Anonymous attackers located in 3 different subnets are launching slow attacks on nodes in the server farm in a random manner as described in section 3.4.1.2. Anomalous traffic, by means of unusual port numbers, is generated along with normal traffic within and between networks. To simulate innocent events like user mistakes suspicious traffic is also generated by normal nodes but at different rates as in section 3.4.1.2. Each simulation is run for a reasonable period of time to ensure that enough traffic is generated (over one million events). Prior probabilities and likelihoods in Equation 6.1 are

assigned as described in section 3.4.1.3. Four cases are considered. In case 1 two nodes on the server farm are targeted by three attackers, in case 2 one node on the server farm is targeted by three attackers, in case 3 one node on the server farm is targeted by a single attacker, and in case 4 two nodes on the server farm are targeted by one attacker.

6.4 Results

This section presents graphs obtained for both approaches using the same trace in each case. Figures 6.1 to 6.4 depict how the targets of slow attacks are detected in all four cases. Graphs in Figures 6.1 to 6.4 are obtained utilising destination information. Hence Equation 6.1 is used. *Min*, *Max* and T are the minimum, maximum and threshold for profile scores of normal nodes in each subnet where a targeted node V (or V_i, i=1,2 in cases 1 and 4) is located.

Figures 6.5 to 6.8 show how attackers are detected in all four cases. These graphs are obtained utilising source information. Therefore Equation 3.4 in Chapter 3 is used for this purpose. *Min*, *Max* and T are the minimum, maximum and threshold for profile scores of normal nodes in each subnet where an attacker node A is located. Since similar results are obtained for all three attackers in cases 1 and 2 only one attacker is presented.

6.5 Discussion

Our approach should not be mistaken as a host based monitoring scheme. The focus of work presented in this chapter is not a problem of location of IDS deployment. It focuses the event analysis stage of a monitoring system.

As mentioned in section 5.3.3.2, detection potential measures how likely an activity could be detected as a suspicious slow activity. Figures 6.9 to 6.12 compares detection potential across two approaches in each case. A (or A_i) represents the detection potential for attackers while V (or V_i) represents the detection potential for targets. The latter has a higher detection potential in all cases. Higher variations (fluctuations) on detection potential indicate a higher chance for false alarms.

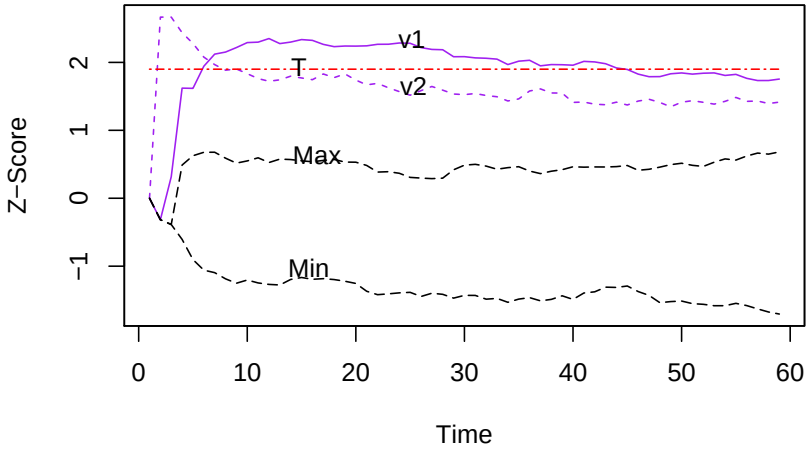

Figure 6.1: Utilising destination information. Case 1 - two targets

.

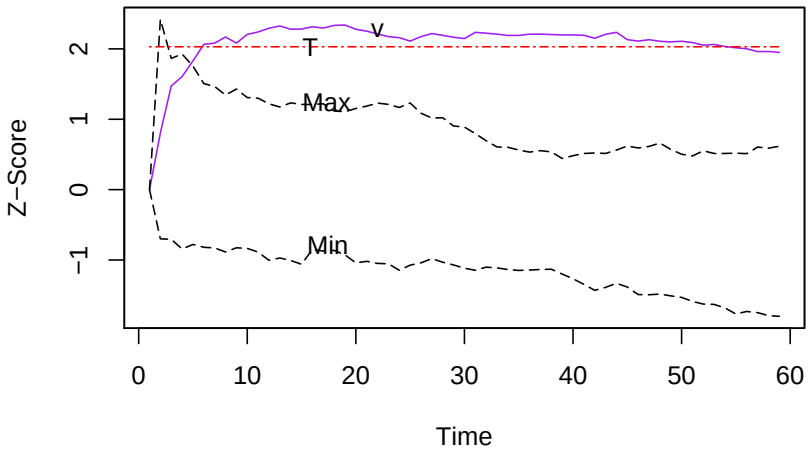

Figure 6.2: Utilising destination information. Case 2 - target

.

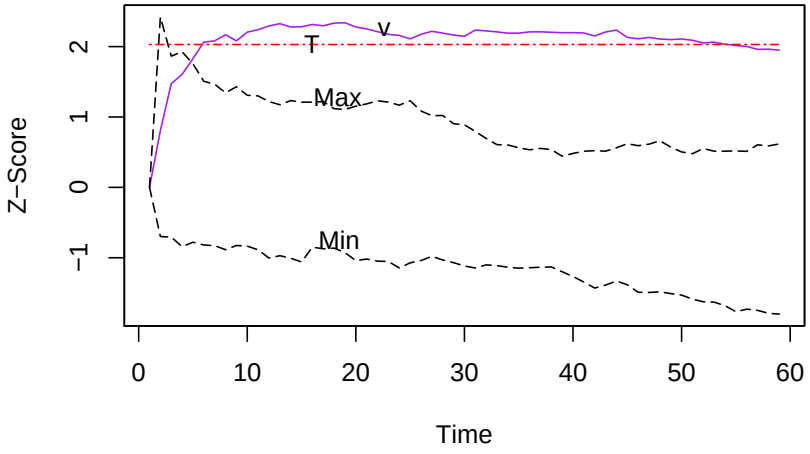

Figure 6.3: Utilising destination information. Case 3 - target

.

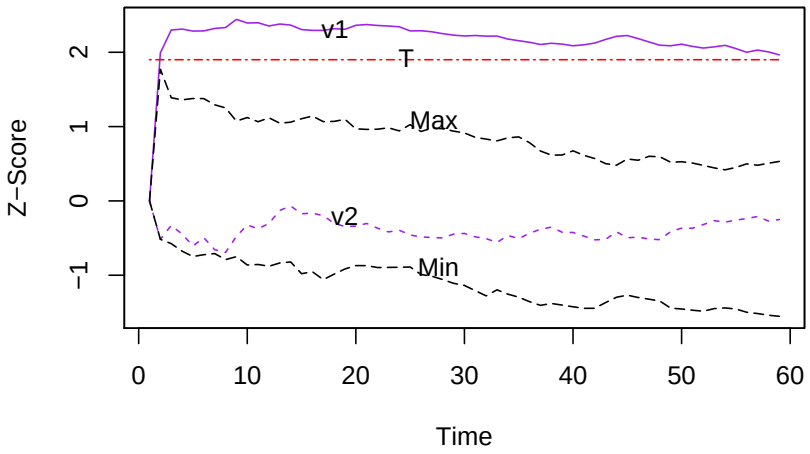

Figure 6.4: Utilising destination information. Case 4 - two targets

.

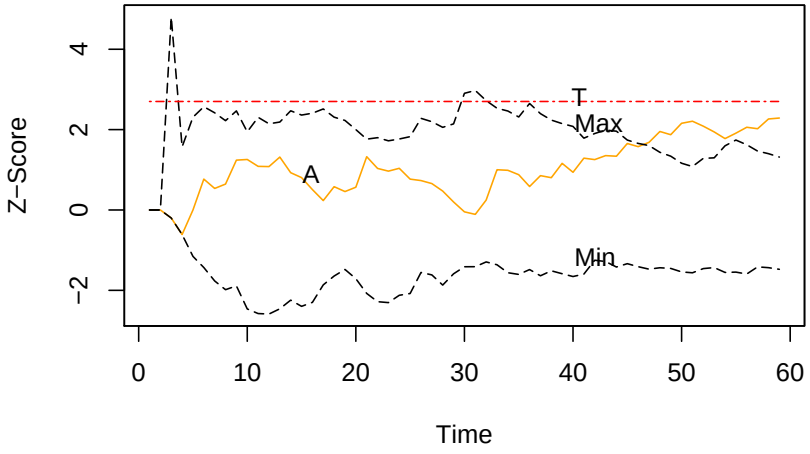

Figure 6.5: Utilising source information. Case 1 - attacker 1

.

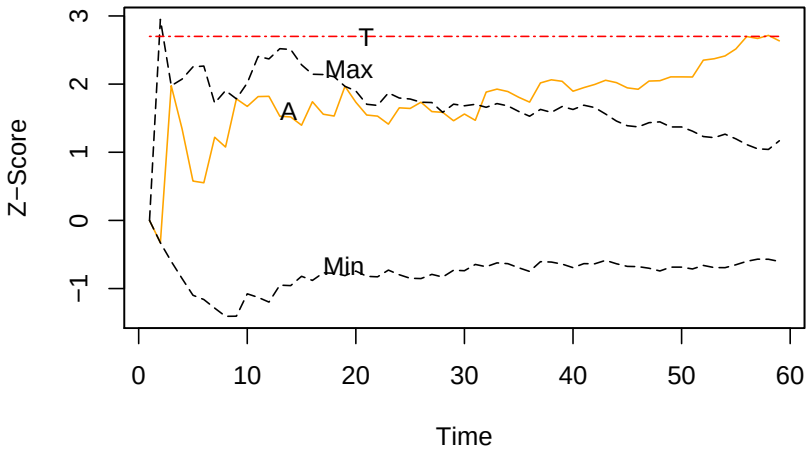

Figure 6.6: Utilising source information. Case 2 - attacker 1

.

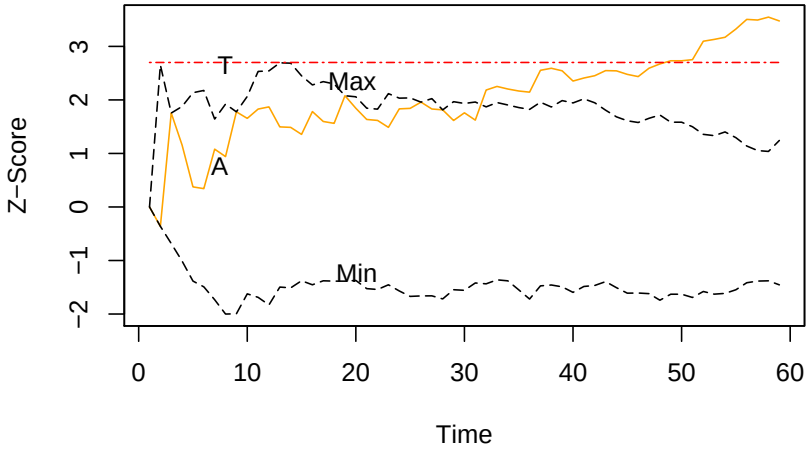

Figure 6.7: Utilising source information. Case 3 - attacker
.

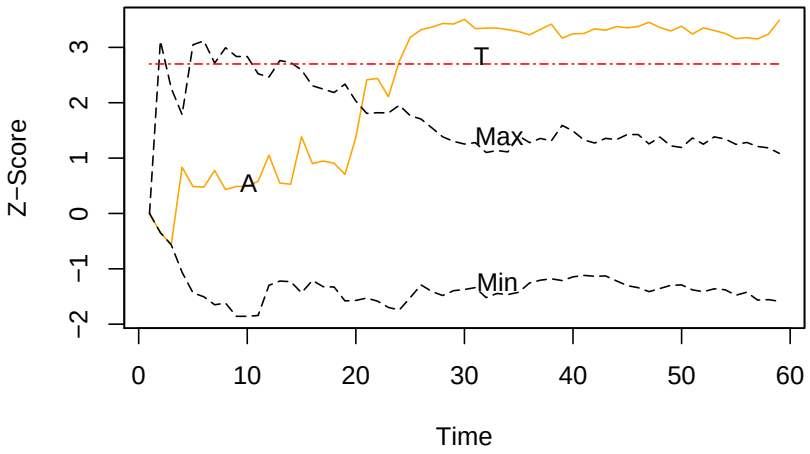

Figure 6.8: Utilising source information. Case 4 - attacker
.

Our approach is capable of detecting targets under attack successfully (see Figures 6.1 to 6.4). Targets cut off (or very close to) the threshold while normal nodes in target's subnet are significantly away from the threshold. As depicted in Figures 6.1 to 6.4 attackers hide among normal nodes and the source-centric approach fails to detect them as quickly. Case 4, where colluded activities are not simulated, is an exception here as it detects only one target out of two. But in case 1, both target nodes are detected; a minimum number of observations are required in order to detect a target successfully. In case 1, since three attackers target two victims, there is a better chance for the monitoring system to observe enough evidence against each victim than it is in case 4. Finding the relationship between detectability and minimum number of observations required is future work.

One difficulty with attribution is that attacks are carried out in multiple stages using compromised machines as stepping stones (or in the form of botnets). One argues that monitoring systems could be deployed to achieve both attribution and early warning for attacks on target nodes. While this is feasible in theory, in practice this means the cost of monitoring is incredibly high, as networks expand in size, traffic volume rise, and slow attackers get slower. In one sense, attribution is not serving effective security, but only a distraction for many network administrators. Make no mistake that attribution remains important but this is best left to be carried out by dedicated Cyber crime units, perhaps operating at regional or national level providing for a coordinated response for potential attribution. Only then are the complexities involved in responding to large-scale organised attacks (Clayton, 2006) could be overcome both technically and otherwise.

6.6 Related work

Whyte *et al.* (2006) offers a different direction for security monitoring by proposing a class of scanning detection algorithms that focus on what is being scanned for instead of who is scanning. But such an approach is not completely independent from the source information either. It uses the source information of scan packets for victim detection. Our approach does not require any information

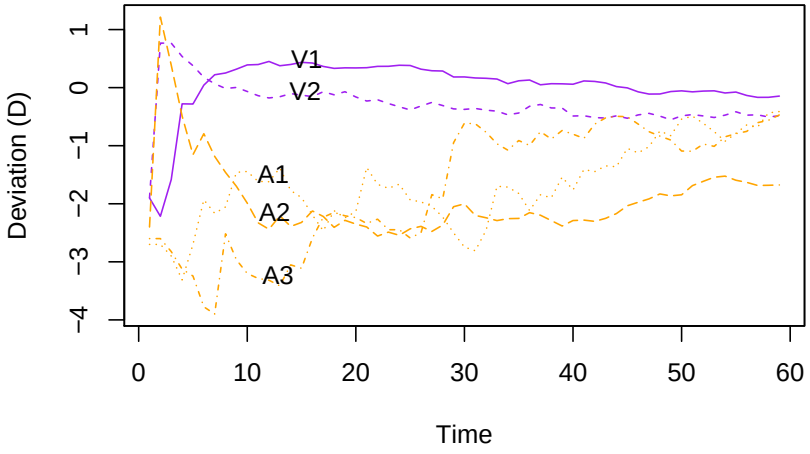

Figure 6.9: Detection potential for case 1

.

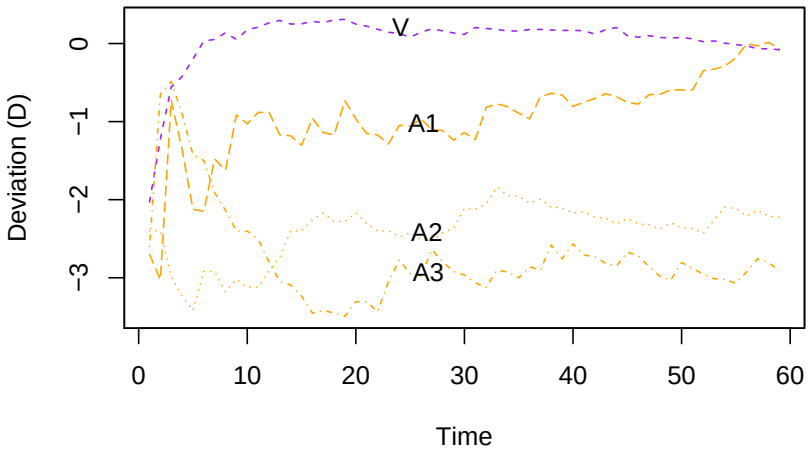

Figure 6.10: Detection potential for case 2

.

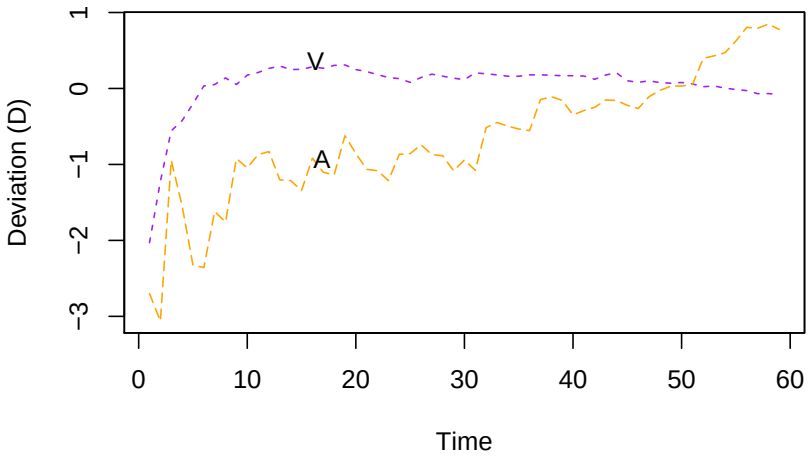

Figure 6.11: Detection potential for case 3

.

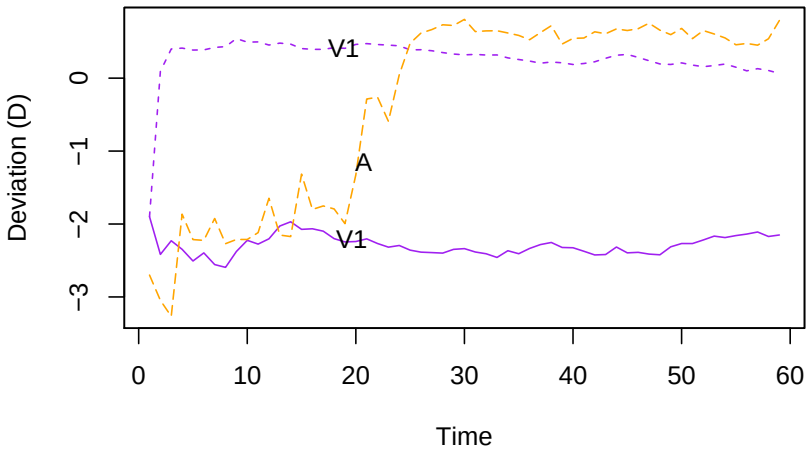

Figure 6.12: Detection potential for case 4

.

about the source. It completely depends on destination information and allows for any suspicious event on the network to be accounted for. Most importantly, we acknowledge two types of uncertainties of events defined in section 3.3.1 in a Bayesian framework. Hence our effort is completely different from (Whyte *et al.*, 2006), but has been inspired from that work. Using Bayesian technique and its variants for intrusion detection can be found in (Siaterlis & Maglaris, 2004). The relevance of information fusion for network security monitoring has been widely discussed (Chandola *et al.*, 2009; Vokorokos *et al.*, 2008).

6.7 Conclusion

We utilise a data fusion algorithm to combine the output of several information sources to a single score. It acts as a data reduction method and enables us to propose a lightweight monitoring scheme for the problem which is essential in near-real-time analysis of slow, sophisticated targeted attacks. Our approach promises scalable means for detection. A case study is presented for demonstration, and experimental results are encouraging. Experimental results offer a promise for the feasibility of target detection in network security monitoring. Target-centric monitoring provides for effective detection of slow and suspicious activities as one does not have to rely on possible source aggregation. This could be particularly useful for monitoring of high-profile nodes that are at particular risk from sophisticated insider attacks or purposefully designed cyber weapons. The focus on targeted nodes takes into account the importance of preventing such compromise, which in itself should help to undermine attacks.

Finally the technical findings of this chapter include:

- proposing a radical shift to the focus of analysis.

Chapter 7

Conclusion

This thesis set out to establish a theoretical framework for an efficient monitoring scheme for slow activities in computer networks. In this final chapter we will review the research contributions of this thesis, as well as discuss directions for future research.

7.1 Contributions

This doctoral research has addressed some important questions in the area of monitoring of slow suspicious activities on computer networks. We revisit the research questions set out in section 1.1 and summarise how each of them has been addressed.

- **A Scalable monitoring scheme** was proposed to fulfil the first objective set out in section 1.1. As mentioned in Chapter 3, our approach is scalable in terms of storage. Organizations find difficulties to weed through the noise of routine security events and determine which threats warrant further investigations. The profiling technique in section 3.3 addresses this issue. It acknowledges the motivation uncertainty and hence reduces possible false alarms. As mentioned in section 3.3.3.1, normal behaviour keeps evolving and a current notion of normal behaviour is not sufficiently representative in the future is a critical issue in this domain. Using Grubbs' test in our work overcomes that issue.

- **The Tracing algorithm** proposed in Chapter 4 also serves for the first objective in section 1.1. It is often the case in the networking world one wishes to know "who did that", and locate the right person responsible with a view of persuading them not to do that again. In this regard, finding the correct origin of the activities is very important. In a situation there are multiple suspected sites to investigate prioritization centres of attention would be a problematic. Tracing algorithm proposed in Chapter 4 serves for this task. It helps to speed up the investigation process and to save the cost and time spending on it.

- **Lightweight monitoring scheme** proposed in Chapter 5 addresses our second objective of this research in section 1.1. We have shown that traffic sampling can be employed with our monitoring algorithm, and some population characteristics (proportion) of network traffic remain unchanged in samples. We hope this analysis helps further advancement of researches on traffic sampling for security specific tasks which currently has drawn comparatively a less attention of research community in this domain.

- **Target centric monitoring scheme** proposed in Chapter 6 addresses final research objective set out in section 1.1. Proposed monitoring scheme completely depends only on information within your control and ignores depending on any source information to protect networks. This work demonstrated the core principle taking into account the importance of preventing such compromise. Moreover, our approaches to tracing the source of such activity and a target-centric method to monitoring offer means to significantly improve network security monitoring against increasing volumes of traffic, spoofing attempts and collusion.

7.2 Future work

Our work opens several research questions for future works. Some of them are briefly described as below.

- **Sophisticated attacker models** - further work should attempt to take

into account more sophisticated attacker models against number of different test cases either in simulated or real world environments. The point of the current work is to demonstrate that the approach in principle is sound and works. Further work should aim to address this particular point.

- **Information fusion technique** - we use a rather simple model for information fusion in this thesis. Investigations to find the optimal method for information fusion among many other available methods should be performed in future.

- **SIEM deployment** - Statistically independence between information sources is a key assumption in the proposed approach. Although this assumption seems reasonable in many situations in SIEM deployment, future study should be performed to identify exceptional cases that violate this condition, and to treat accordingly for such cases.

- **Classification algorithms** - As mentioned in section 3.5.4.3, doing a peer analysis within a similar peer group would give a better detection accuracy in terms of lower false alarm rates. Future study should be performed to introduce a suitable classification algorithm to this work in order to reduce possible false alarms.

7.2.1 Implementation

Implementation of proposed approaches for real world networks is essential for studying the requirements of host statistics such as CPU load, I/O operations and memory usage in its runtime. This implementation can be done as a separate monitoring system or as a component of existing monitoring system such as Snort IDS.

7.2.2 Evaluation against real world

Most widely employed data sets for IDSs evaluation are DARPA and KDD99 which was created in 1998 and 1999. Still some researches use these two data sets

to evaluate IDSs while others criticise them. Traffic for both data sets have been generated via simulating the attacks and the background traffic and focusing on rapid attacks, and hence not suitable for evaluation of slow attacks. During last 15 years there is no prominent IDS evaluation data set has been emergence, instead most researches rely on simulation approach. This is evident that how difficult to testing a new approach in this domain against real world data. Some data sets like LBN, ISCX2012 have been produced, but lack of attention is drawn from the research community. Therefore, one possible option is using a community-lab test bed which will allow testing the research result under realistic conditions and learn from it to improve the idea. This would be a future step of this research.

Appendix A

1. ***WEB-MISC robots.txt access***

   Attack Scenario[1]

   An attacker could retrieve robots.txt (made by search robots for site indexing) from the server, and discover the path to an unprotected administration interface for the server, and then gain control of the web-server using this interface.

   A Snort Rule [2]

   ```
   alert tcp $external_net any -> $http_servers $http_ports
   (msg:"web-misc robots.txt access"; flow:to_server,
   established; uricontent:"/robots.txt"; nocase; reference:
   nessus,10302;classtype:web-application-activity;threshold:
   type threshold, track by_dst, count 10, seconds 60;
   sid:1000852; rev:1;)
   ```

   Discussion

   An event is generated when an attempt is made to access the file robots.txt directly. The rule logs every 10th event on this SID during a 60 second interval. So if less than 10 events occurred in 60 seconds, nothing gets logged.

[1] Source:http://www.selfsecurity.org/TrendMap/signature/eng/8.htm
[2] Source:http://manual.snort.org/node35.html

Appendix B

1. **R code for node score calculation for peer analysis.**

```
library(sqldf)
nodeScore # A table to hold node scores during a very long time window W
networkName # Network name where attacker node is located
networkTrace # A trace observed during a very smaller time window w
portNumbers # Set of usual and unusual port numbers with their likelihoods
priorBelief # Prior belief of each node being in states H1 and H2
substrRight <- function(x, n){substr(x, nchar(x)-n+1, nchar(x))} # A function def~
nodelist<-as.data.frame(sqldf(paste("Select distinct c5 from ",_
networkTrace," where c5 like ","'",networkName,"'",sep = ""))) # Events from a network
nodeScore<- as.table(nodeScore)
for (i in 1:ncol(nodeScore)){
eventList<-as.data.frame(sqldf(paste("Select distinct c3,c7 from ",_
networkTrace," where c5 like ","'",colnames(nodeScore)[i],"'", _
" order by c3",sep = ""))) # Events from a particular node
if (nrow(eventList)>0) {
tempScoreH1=1
tempScoreH2=1
for (j in 1:nrow(eventList)){
for (k in 1:ncol(portNumbers)){
if (substrRight(toString(eventList[j,2]),5)==colnames(portNumbers)[k]){
tempScoreH1=tempScoreH1*portNumbers[1,colnames(portNumbers)[k]]
tempScoreH2=tempScoreH2*portNumbers[2,colnames(portNumbers)[k]]}}}
postProb=tempScoreH1*priorBelief[1,colnames(nodeScore)[i]]/(tempScoreH1*priorBelief[1,_
colnames(nodeScore)[i]]+tempScoreH2*priorBelief[2,colnames(nodeScore)[i]])
nodeScore[1,colnames(nodeScore)[i]]=nodeScore[1,colnames(nodeScore)[i]]+postProb}}
nodeZscore[1,]=scale(nodeScore[1,]) # Converting node scores to Z-Scores
write.table(nodeZscore, paste("nodeZscore_",networkName,".csv",sep = ""), _
sep=",",col.names=T,row.names=T,quote=F,append = TRUE) # Optional, if needed _
#to maintain score table on external storage
plot(nodeZscore[,1],col="white",xlab="Time", ylab = "Z-Score",cex=0.75, _
ylim=range(nodeScore),cex.lab=0.75,cex.axis=0.75) # Plotting graphs for each node
for (m in 1:ncol(nodeZscore)){
lines(nodeZscore[,m], col="orange",lty=m)}
threshold=grubbs.test(nodeZscore, type = 20, opposite = FALSE, two.sided = FALSE)
lines(threshold, col="red",lty=6)
box()
```

2. **R code for anomaly score calculation for discord analysis.**

```
library(sqldf)
library(forecast)
nodeScore # Load the Node score table created by peer analysis program above
for (i in 1:ncol(nodeScore)){
ln=25 # Length of the ARIMA model to be used
totdv=0 # Total deviation from CIs
totpeak=0 # Total number of cut off points
rst <- array(1:length(nodeScore[,i]), dim=c(length(nodeScore[,i]),8)) # An array to_
# store predicted CIs and acctual values
rst[,1]=nodeScore[,i] # Acctual values
rst[,2]=nodeScore[,i] # Acctual values
rst[,3]=0.0 # To store fitted values
rst[,4]=0.0 # To store upper bound of 95%CI
rst[,5]=0.0 # To store upper bound of 80%CI
rst[,6]=0.0 # To store lower bound of 80%CI
rst[,7]=0.0 # To store lower bound of 95%CI
rst[,8]=0.0 # number of times acctual cut off the CI
for (j in (ln+1):(length(nodeScore[,i])-2))
{
xtf<-auto.arima(rst[(j-ln):(j-1),2]) # Forecasting next possible value with CIs
fv<-forecast(xtf,1)
dffv<-data.frame(fv)
rst[j,3]<-dffv$Point.Forecast[1]
rst[j,4]<-dffv$Hi.95
rst[j,5]<-dffv$Hi.80
rst[j,6]<-dffv$Lo.80
rst[j,7]<-dffv$Lo.95
if (rst[j,2]>rst[j,4]) # Acctual value cuts off the Upper bound of 95%CI
{
rst[j,8]=1
totpeak=totpeak+1
totdv=totdv+abs(rst[j,2]-rst[j,4])
}
if (rst[j,2]<rst[j,7])  # Acctual value cuts off the lower bound of 95%CI
{
totpeak=totpeak+1
rst[j,8]=1
totdv=totdv+abs(rst[j,2]-rst[j,7])
}
if (totpeak>1){
summrdv[j,i]=totdv/(totpeak-1)} # Anomaly score for discord analysis
}
}  # End of the loop
write.table(summrdv, paste("summrdv_",networkName,".csv",sep = ""), _
sep=",",col.names=T,row.names=T,quote=F,append = TRUE) # Optional, if needed _
#to maintain anomaly score table on external storage
plot(summrdv[,1],col="white",xlab="Time", ylab = "Z-Score",cex=0.75, _
ylim=range(nodeScore),cex.lab=0.75,cex.axis=0.75) # Plotting graphs for each node
for (m in 1:ncol(summrdv)){
```

```
lines(summrdv[,m], col="orange",lty=m)}
threshold=grubbs.test(summrdv, type = 20, opposite = FALSE, two.sided = FALSE)
lines(threshold, col="red",lty=6)
box()
```

3. **C++ code in NS3 for simulating the network topology and attackers.**

```
//This program was written to simulate an attacker sending packets to three victims.
//The same prgram can be simply edited to
//simulate all other experiments presented in this thesis.
//This code segment developed by customising
//an example code given in NS3.
// This script exercises global routing code in
//a mixed point-to-point and csma/cd environment
#include <iostream>
#include <fstream>
#include <string>
#include <cassert>
#include "ns3/core-module.h"
#include "ns3/simulator-module.h"
#include "ns3/node-module.h"
#include "ns3/helper-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("MixedGlobalRoutingExample");
int
main (int argc, char *argv[]){
  Config::SetDefault ("ns3::OnOffApplication::PacketSize",_
 UintegerValue (210));
  Config::SetDefault ("ns3::OnOffApplication::DataRate", _
StringValue ("448kb/s"));
  CommandLine cmd;
  cmd.Parse (argc, argv);
  NS_LOG_INFO ("Create nodes.");
  NodeContainer c;
  c.Create (216);
  uint32_t k = p //subtitute p with different
  // numbers to obtain different size subnets
  uint32_t t = q //run time of the simulation
  uint16_t port1 = r;// choose legitimate ports randomly
  // for r to generate normal packets
  uint16_t port2 = s;// choose unusual ports randomly
  // for s to generate suspicious packets
  uint16_t r1 = onrate;// on time rate for packet generating
  uint16_t r2 = offrate;// off time rate for packet generating
  uint16_t s =0;// applications stating times, assign sutable values accordingly
  uint16_t t =3600;// applications ending times, assign sutable values accordingly
  NodeContainer lan11;//node container for subnet 1.1
 for (uint32_t i = 0; i < 25; ++i)
  {
      lan11.Add(c.Get (i));
  }
```

```
      lan11.Add(c.Get (101));
NodeContainer lan121;//node container for subnet 1.2.1
 for (uint32_t i = 25; i < 35; ++i)
  {
      lan121.Add(c.Get (i));
  }
      lan121.Add(c.Get (101));
NodeContainer lan122;//node container for subnet 1.2.2
 for (uint32_t i = 35; i < 50; ++i)
  {
      lan122.Add(c.Get (i));
  }
      lan122.Add(c.Get (101));
 NodeContainer lan21;//node container for subnet 2.1
 for (uint32_t i = 50; i < 65 ; ++i)
  {
      lan21.Add(c.Get (i));
  }
      lan21.Add(c.Get (101));
NodeContainer lan221;//node container for subnet 2.2.1
 for (uint32_t i = 65; i < 85 ; ++i)
  {
      lan221.Add(c.Get (i));
  }
      lan221.Add(c.Get (101));
NodeContainer lan222;//node container for subnet 2.2.2
 for (uint32_t i = 85; i < 100 ; ++i)
  {
      lan222.Add(c.Get (i));
  }
      lan222.Add(c.Get (101));
 NodeContainer serv;//node container for server farm
 for (uint32_t i = 106; i < 116 ; ++i)
  {
      serv.Add(c.Get (i));
  }
     serv.Add(c.Get (102));
 NodeContainer lan311;//node container for subnet 3.1.1
 for (uint32_t i = 116; i < 126 ; ++i)
  {
      lan311.Add(c.Get (i));
  }
     lan311.Add(c.Get (102));
NodeContainer lan312;//node container for subnet 3.1.2
 for (uint32_t i = 126; i < 136 ; ++i)
  {
      lan312.Add(c.Get (i));
  }
     lan312.Add(c.Get (102));
 NodeContainer lan321;//node container for subnet 3.2.1
```

```
 for (uint32_t i = 136; i < 151 ; ++i)
  {
      lan321.Add(c.Get (i));
  }
     lan321.Add(c.Get (102));
NodeContainer lan322;//node container for subnet 3.2.2
 for (uint32_t i = 151; i < 166 ; ++i)
  {
      lan322.Add(c.Get (i));
  }
     lan322.Add(c.Get (102));
NodeContainer lan4;//node container for subnet 4
 for (uint32_t i = 166; i < 216 ; ++i)
  {
      lan4.Add(c.Get (i));
  }
     lan4.Add(c.Get (102));
 NodeContainer unal;//node container for unallocated core
 for (uint32_t i = 100; i < 106 ; ++i)
  {
      unal.Add(c.Get (i));
  }
  InternetStackHelper internet;
  internet.Install (c);
  // We create the channels first without
  //any IP addressing information
  NS_LOG_INFO ("Create channels.");
  CsmaHelper csma;
  csma.SetChannelAttribute ("DataRate", StringValue ("5Mbps"));
  csma.SetChannelAttribute ("Delay", StringValue ("2ms"));
  NetDeviceContainer dvc11 = csma.Install (lan11);
  NetDeviceContainer dvc121 = csma.Install (lan121);
  NetDeviceContainer dvc122 = csma.Install (lan122);
  NetDeviceContainer dvc21 = csma.Install (lan21);
  NetDeviceContainer dvc221 = csma.Install (lan221);
  NetDeviceContainer dvc222 = csma.Install (lan222);
  NetDeviceContainer dvcSr = csma.Install (serv);
  NetDeviceContainer dvc311 = csma.Install (lan311);
  NetDeviceContainer dvc321 = csma.Install (lan321);
  NetDeviceContainer dvc312 = csma.Install (lan312);
  NetDeviceContainer dvc322 = csma.Install (lan322);
  NetDeviceContainer dvc4= csma.Install (lan4);
  NetDeviceContainer dvunal = csma.Install (unal);
  // Later, we add IP addresses.
  NS_LOG_INFO ("Assign IP Addresses.");
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  ipv4.Assign (dvc11);
  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
  ipv4.Assign (dvc121);
```

```
ipv4.SetBase ("10.1.3.0", "255.255.255.0");
ipv4.Assign (dvc122);
ipv4.SetBase ("10.1.4.0", "255.255.255.0");
ipv4.Assign (dvc21);
ipv4.SetBase ("10.1.5.0", "255.255.255.0");
ipv4.Assign (dvc221);
ipv4.SetBase ("10.1.6.0", "255.255.255.0");
ipv4.Assign (dvc222);
ipv4.SetBase ("10.1.7.0", "255.255.255.0");
ipv4.Assign (dvc311);
ipv4.SetBase ("10.1.8.0", "255.255.255.0");
ipv4.Assign (dvc321);
ipv4.SetBase ("10.1.9.0", "255.255.255.0");
ipv4.Assign (dvc312);
ipv4.SetBase ("10.1.10.0", "255.255.255.0");
ipv4.Assign (dvc322);
ipv4.SetBase ("10.1.11.0", "255.255.255.0");
ipv4.Assign (dvc4);
ipv4.SetBase ("10.1.12.0", "255.255.255.0");
Ipv4InterfaceContainer i5i6 = ipv4.Assign (dvcSr);
ipv4.SetBase ("10.250.1.0", "255.255.255.0");
ipv4.Assign (dvunal);
// Create router nodes, initialize routing database and set up the routing
// tables in the nodes.
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
//innocent events generate to first node in serverfarm
NS_LOG_INFO ("Create Applications.");
OnOffHelper onoff1 ("ns3::UdpSocketFactory", Address (InetSocketAddress -
(Ipv4Address ("10.1.12.2"), port1)));
onoff1.SetAttribute ("OnTime", RandomVariableValue (ExponentialVariable (r1)));
onoff1.SetAttribute ("OffTime", RandomVariableValue (ExponentialVariable (r2)));
onoff1.SetAttribute ("DataRate", StringValue ("300bps"));
onoff1.SetAttribute ("PacketSize", UintegerValue (50));
ApplicationContainer app1;
for (uint32_t i = 0; i < 216 ; ++i)
{
     app1.Add(onoff1.Install (c.Get (i)));
}
app1.Start (Seconds (s));
app1.Stop (Seconds (t));//need to t=3600 to achieve the same number of events
//as in the paper innocent events generate to other nodes in serverfarm
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address -
("10.1.12.1"), port1)));
ApplicationContainer app2;
for (uint32_t i = 0; i < 216 ; ++i)
{
     app2.Add(onoff1.Install (c.Get (i)));
}
app2.Start (Seconds (s));
app2.Stop (Seconds (t));
```

```
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
(Ipv4Address ("10.1.12.3"), port1)));
ApplicationContainer app32;
for (uint32_t i = 0; i < 216 ; ++i)
{
     app32.Add(onoff1.Install (c.Get (i)));
}
app32.Start (Seconds (s));
app32.Stop (Seconds (t));
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
(Ipv4Address ("10.1.12.4"), port1)));
ApplicationContainer app42;
 for (uint32_t i = 0; i < 216 ; ++i)
{
     app42.Add(onoff1.Install (c.Get (i)));
}
app42.Start (Seconds (s));
app42.Stop (Seconds (t));
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
(Ipv4Address ("10.1.12.5"), port1)));
ApplicationContainer app52;
for (uint32_t i = 0; i < 216 ; ++i)
{
     app52.Add(onoff1.Install (c.Get (i)));
}
app52.Start (Seconds (s));
app52.Stop (Seconds (t));
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
(Ipv4Address ("10.1.12.6"), port1)));
ApplicationContainer app62;
for (uint32_t i = 0; i < 216 ; ++i)
{
     app62.Add(onoff1.Install (c.Get (i)));
}
app62.Start (Seconds (s));
app62.Stop (Seconds (t));
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
(Ipv4Address ("10.1.12.7"), port1)));
ApplicationContainer app72;
for (uint32_t i = 0; i < 216 ; ++i)
{
     app72.Add(onoff1.Install (c.Get (i)));
}
app72.Start (Seconds (s));
app72.Stop (Seconds (t));
onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
(Ipv4Address ("10.1.12.8"), port1)));
ApplicationContainer app82;
for (uint32_t i = 0; i < 216 ; ++i)
{
```

```
      app82.Add(onoff1.Install (c.Get (i)));
  }
  app82.Start (Seconds (s));
  app82.Stop (Seconds (t));
  onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
  (Ipv4Address ("10.1.12.9"), port1)));
  ApplicationContainer app92;
  for (uint32_t i = 0; i < 216 ; ++i)
  {
      app92.Add(onoff1.Install (c.Get (i)));
  }
  app92.Start (Seconds (s));
  app92.Stop (Seconds (t));
  onoff1.SetAttribute ("Remote", AddressValue (InetSocketAddress _
  (Ipv4Address ("10.1.12.10"), port1)));
  ApplicationContainer app102;
  for (uint32_t i = 0; i < 216 ; ++i)
  {
      app102.Add(onoff1.Install (c.Get (i)));
  }
  app102.Start (Seconds (s));
  app102.Stop (Seconds (t));
  //configure suspicious events, generates to one node_
 //in server farm.
  OnOffHelper onoff2 ("ns3::UdpSocketFactory", Address (InetSocketAddress _
  (Ipv4Address ("10.1.12.2"), port2)));//victim 1
  onoff2.SetAttribute ("OnTime", RandomVariableValue (ExponentialVariable (r1)));
  onoff2.SetAttribute ("OffTime", RandomVariableValue (ExponentialVariable (r2)));
  onoff2.SetAttribute ("DataRate", StringValue ("300bps"));
  onoff2.SetAttribute ("PacketSize", UintegerValue (50));
  ApplicationContainer app3,app5,app7;
  app3.Add(onoff2.Install (c.Get (2)));
  app3.Start (Seconds (s));
  app3.Stop (Seconds (t));
  app5.Add(onoff2.Install (c.Get (67)));
  app5.Start (Seconds (s));
  app5.Stop (Seconds (t));
  app7.Add(onoff2.Install (c.Get (138)));
  app7.Start (Seconds (t));
  app7.Stop (Seconds (t));
  //for victim 2
  onoff2.SetAttribute ("Remote", AddressValue (InetSocketAddress (Ipv4Address _
("10.1.12.3"), port2))); //victim 2
  ApplicationContainer app4,app6,app8;
  app4.Add(onoff2.Install (c.Get (2)));
  app4.Start (Seconds (100.0));
  app4.Stop (Seconds (t));
  app6.Add(onoff2.Install (c.Get (67)));
  app6.Start (Seconds (140.1));
  app6.Stop (Seconds (t));
```

```
  app8.Add(onoff2.Install (c.Get (138)));
  app8.Start (Seconds (s));
  app8.Stop (Seconds (t));
  //packet capturing in promiscuous mode.
  csma.EnablePcap ("victimProfiling",dvcSr.Get (1), true);
  NS_LOG_INFO ("Run Simulation.");
  Simulator::Run ();
  Simulator::Destroy ();
  NS_LOG_INFO ("Done.");
}
```

4. **R code for selection of samples.**

```
networkName # Network name where attacker node is located
networkTrace # A trace observed during a very smaller time window w
samplingRate # Sampling rate for the whole trace
statumRates # Sampling rates computed for each stratum using Equation 5.1_
#against possible types of network traffic, e.g. colnames(nodeScore) <- c('UDP', 'ICMP','ARP')
networkTrace <- as.table(networkTrace)
library(sqldf)
for (i in 1:ncol(statumRates))
{
tempdata<-sqldf(paste("Select * from ",networkTrace," where Protocol =='",_
colnames(statumRates)[i],"'",sep = ""))
sampleSize=round(nrow(tempdata)*statumRates[1,colnames(statumRates)[i]],0)
smallSample[i] <- tempdata[sample(1:nrow(tempdata),sampleSize,replace=FALSE),]
bigSample <- rbind(smallSample[i],bigSample)
}
write.table(bigSample, paste("bigSample",samplingRate,".csv",sep = ""), _
sep=",", col.names=F, quote=F,append = TRUE)
```

References

Ali, S., Haq, I., Rizvi, S., Rasheed, N., Sarfraz, U., Khayam, S. & Mirza, F. (2010). On mitigating sampling-induced accuracy loss in traffic anomaly detection systems. In *SIGCOMM Computer Communication*, 4–16. 78

Anderson, J.P. (1980). Computer security threat monitoring and surveillance. technical report. Tech. rep., James P. Anderson Co. Fort Washington, Pennsylvania. 5

Androulidakis, G. & Papavassiliou, S. (2008). Improving network anomaly detection via selective flow-based sampling. *Communications, IET*, **2**, 399–409. 78

Androulidakis, G., Chatzigiannakis, V. & Papavassiliou, S. (2009). Network anomaly detection and classification via opportunistic sampling. *Netwrking Magazine of Global Internet working*, **23**, 6–12. 78

Anscombe, F.J. & Guttman, I. (1960). Rejection of outliers . *Technometrics 2*, **2**, 123–147. 34

Ansel, H.C. & Prince, S.J. (2004). *Pharmaceutical calculations: the pharmacist's handbook*. Lippincott Williams & Wilkins. 81

Argus (2012). Argus, the network audit record generation and utilization system. `http://www.qosient.com/argus/`. 27

Ball, R., Fink, G. & North, C. (2004). Home-centric visualization of network traffic for security administration. In *Proc. of the 2004 ACM Workshop on visualization and Data Mining for Computer Security*, 55–64. 18, 19

Barakat, C., Iannaccone, G. & Diot, C. (2005). Ranking flows from sampled traffic. In *Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, 188–199, ACM. 78

Barbara, D., Wu, N. & Jajodia, S. (2001). Detecting novel network intrusions using bayes estimators. In *First SIAM Conference on Data Mining*, Citeseer. 27

Bartos, K. & Rehak, M. (2012). Towards efficient flow sampling technique for anomaly detection. In *TMA 2012 Conference Proceedings, LNCS 7189*, 93–106, Springer-Verlag Berlin Heidelberg. 78

Basu, R., Cunningham, R.K., Webster, S.E. & Lippmann, R.P. (2001). Detecting low-profile probes and novel denial-of-service attacks. Tech. rep., IEEE SMC IAS Workshop 2001, West Point, New York, USA. 26

Beidleman, S.W. (2009). Defining and deterring cyber war. Tech. rep., DTIC Document. 60

Berk, V.H., Cybenko, G., Souza, I.G.d. & Murphy, J.P. (2012). Managing malicious insider risk through bandit. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, 2422–2430, IEEE. 35

Bhuyan, M.H., Bhattacharyya, D. & Kalita, J.K. (2011). Survey on incremental approaches for network anomaly detection. *International Journal of Communication Networks & Information Security*, **3**. 12, 13, 14, 17, 20, 21, 22, 23, 32, 39

Bradford, P.G., Brown, M., Self, B. & Perdue, J. (2004). Towards proactive computer system forensics. In *International conference on information technology: Coding and computing,IEEE Computer Society*. 26

Brauckhoff, D., Tellenbach, B., Wagner, A., May, M. & Lakhina, A. (2006). Impact of packet sampling on anomaly detection metrics. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC '06, 159–164, ACM, New York, NY, USA. 78, 79

Braun, L., Didebulidze, A., Kammenhuber, N. & Carle, G. (2010). Comparing and improving current packet capturing solutions based on commodity hardware. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 206–217, ACM. 74

Bronstein, A., Das, J., Duro, M., Friedrich, R., Kleyner, G., Mueller, M., Singhal, S. & Cohen, I. (2001). Self-aware services: Using bayesian networks for detecting anomalies in internet-based services. In *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, 623–638, IEEE. 27

Brown, C., Cowperthwaite, A., Hijazi, A. & Somayaji, A. (2009). Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, 1–7, IEEE. 21

Brugger, S.T. & Chow, J. (2007). An assessment of the darpa ids evaluation dataset using snort. *UCDAVIS department of Computer Science*, **1**, 2007. 22

Brynielsson, J., Horndahl, A., Johansson, F., Kaati, L., Mårtenson, C. & Svenson, P. (2013). Harvesting and analysis of weak signals for detecting lone wolf terrorists. *Security Informatics*, **2**, 11. 31

Bu, Y., Leung, O.T.W., Fu, A.W.C., Keogh, E.J., Pei, J. & Meshkin, S. (2007). Wat: Finding top-k discords in time series database. In *SDM*, SIAM. 36

Burbeck, K. & Nadjm-Tehrani, S. (2007). Adaptive real-time anomaly detection with incremental clustering. *information security technical report*, **12**, 56–67. 14, 17, 20

Burch, H. & Cheswick, B. (2000). Tracing Anonymous Packets to Their Approximate Source. In *Proc. 2000 of USENIX LISA Conference*. 60, 61

CERT Network Situational Awareness Team (2012). Silk, the system for internet-level knowledge. http://tools.netsa.cert.org/silk. 27

Chandola, V., Banerjee, A. & Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, **41**, 15:1–15:58. 12, 13, 14, 21, 33, 34, 35, 36, 102

Charney, S. (2009). Rethinking the cyber threat: A framework and path forward. *Microsoft Corporation*. 60

Chatfield, C. (2003). *The analysis of time series: an introduction*. CRC press. 37

Chivers, H., Nobles, P., Shaikh, S.A., Clark, J.A. & Chen, H. (2009). Accumulating evidence of insider attacks. In *The 1st International Workshop on Managing Insider Security Threats (MIST 2009)*, 34, Citeseer. 15, 26, 27, 28, 42, 61

Chivers, H., Clark, J.A., Nobles, P., Shaikh, S.A. & Chen, H. (2013). Knowing who to watch: Identifying attackers whose actions are hidden within false alarms and background noise. *Information Systems Frontiers*, **15**, 17–34. 15, 26, 27, 28, 29, 61, 78

Cisco (2013). Cisco netflow. http://www.cisco.com/warp/public/732/Tech/netflow. 77, 78, 79

Claffy, K.C., Polyzos, G.C. & Braun, H.W. (1993). Application of sampling methodologies to network traffic characterization. In *Conference proceedings on Communications architectures, protocols and applications*, SIGCOMM '93, 194–203, ACM, New York, NY, USA. 78, 79

Clauset, A. (2011). Inference, Models and Simulation for Complex Systems. http://tuvalu.santafe.edu/~aaronc/courses/7000/csci7000-001_2011_L0.pdf. 31

Clayton, R. (2006). Complexities in criminalising denial of service attacks. http://www.lightbluetouchpaper.org/2006/02/15/complexities-in-criminalising-denial-of-service-attacks/. 99

CSIEM (2013). Cisco security information event management deployment guide. http://www.cisco.com. 9, 26, 32

DAMBALLA (2013). Advanced Persistent Threats (APT). https://www.damballa.com/knowledge/advanced-persistent-threats.php. 11

Damien Miller (2012). Softflowd, flow-based network traffic analyser. http://www.mindrot.org/projects/softflowd/. 27

Das, K. & Schneider, J. (2007). Detecting anomalous records in categorical datasets. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 220–229, ACM. 27

Daskalakis, C., Diakonikolas, I. & Servedio, R.A. (2012). Learning poisson binomial distributions. In *Proceedings of the 44th symposium on Theory of Computing*, 709–728, ACM. 35

Davidoff, S. & Ham, J. (2012). *Network Forensics: Tracking Hackers Through Cyberspace*. Prentice Hall. 16, 27, 29, 42, 61

De Tangil Rotaeche, G.S., Palomar-González, E., Ribagorda-Garnacho, A. & Ramos-Álvarez, B. (2010). Anonymity in the service of attackers. *Serbian Publication InfoReview joins UPENET, the Network of CEPIS Societies Journals and Magazines*, 27–30. 29, 59

Delooze, L. & Kalita, J. (2006). applying soft computing techniques to intrusion detection. In *Proc. of Cyber Security and Information Infrastructure Research Workshop, Oak Ridge National Laboratory, Oak Ridge, TN*. 12, 22

Denning, D.E. (1987). An intrusion-detection model. *Software Engineering, IEEE Transactions on*, 222–232. 6

Diaz-Garcia, J.A. & Cortez, L.U. (2006). Optimum allocation in multivariate stratified sampling: multi-objective programming. *Comunicación Técnica*

No. I-06-07/28-03-206 (PE/CIMAT). Guanajuato, México: Centro de Investigación en Matemáticas, AC. 80

Drew, S. (n.d). Intrusion Detection FAQ: What is the Role of Security Event Correlation in Intrusion Detection? http://www.sans.org/security-resources/idfaq/role.php. 30, 60

Duffield, N. (2004). Sampling for passive internet measurement: A review. *Statistical Science*, **19**, 472–498. 77

Duffield, N., Lund, C. & Thorup, M. (2002). Properties and prediction of flow statistics from sampled packet streams. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, 159–171. 78

Duffield, N., Lund, C. & Thorup, M. (2005). Estimating flow distributions from sampled flow statistics. *IEEE/ACM Transactions on Networking*, **13**, 933–946. 78

Eberle, W., Graves, J. & Holder, L. (2010). Insider threat detection using a graph-based approach. *Journal of Applied Security Research*, **6**, 32–81. 16

Eldardiry, H., Bart, E., Liu, J., Hanley, J., Price, B. & Brdiczka, O. (2013). Multi-domain information fusion for insider threat detection. In *2013 IEEE Security and Privacy Workshops*. 35

Fazio, P., Tan, K. & Kotz, D. (2012). Effects of network trace sampling methods on privacy and utility metrics. In *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS)*, 1–8. 78

Fisk, M., Smith, S., Weber, P., Kothapally, S. & Caudell, T. (2003). Immersive network monitoring. In *proc. PAM2003 Passive and Active Measurement 2003*. 18, 19

Fu, A.W.C., Leung, O.T.W., Keogh, E. & Lin, J. (2006). Finding time series discords based on haar transform. In *Advanced Data Mining and Applications*, 31–41, Springer. 36

GeNIe (n.d). GeNIe-Documentation, Decision Theoritic Modelling: Probability. http://genie.sis.pitt.edu/wiki/Decision-Theoritic-Modelling:-Probability. 42

Giacinto, G. & Roli, F. (2002). Intrusion detection in computer networks by multiple classifier systems. In *Proc. of International Conference on Pattern Recognition. Los Alamitos, CA*. 12, 21

Gonzalez, J.M., Paxson, V. & Weaver, N. (2007). Shunting: a hardware/-software architecture for flexible, high-performance network intrusion prevention. In *Proceedings of the 14th ACM conference on Computer and communications security*, 139–149. 75

Greitzer, F., Paulson, P., Kangas, L., Edgar, T., Zabriskie, M., Franklin, L. & Frincke, D. (2009). Predictive modelling for insider threat mitigation, pacific northwest national laboratory, richland, wa, tech. rep. pnnl technical report. 15, 16

GRUBBS, R.E. (1969). Procedures for Detecting Outlying Observations in Samples. *Technometrics*, **11**, 1–21. 34

Guo, H. (2011). A simple algorithm for fitting a gaussian function [dsp tips and tricks]. *Signal Processing Magazine, IEEE*, **28**, 134–137. 34

Hackers (2013). Slowloris http dos. http://ha.ckers.org/slowloris/. 2, 11

Hagen, N., Kupinski, M. & Dereniak, E.L. (2007). Gaussian profile estimation in one dimension. *Applied optics*, **46**, 5374–5383. 34

Hall, D.D.L. & McMullen, S.A.H. (2004). *Mathematical Techniques in Multisensor Data Fusion 2nd Ed.*. Artech House Publishers. 57

Hall, D.L. & Garga, A.K. (1999). Pitfalls in data fusion (and how to avoid them). In *Proceedings of the Second International Conference on Information Fusion (Fusion99)*, vol. 1, 429–436. 57

Hall, D.L. & Llinas, J. (1997). An introduction to multisensor data fusion. *Proceedings of the IEEE*, **85**, 6–23. 57

Hall, D.L. & Llinas, J. (2001). *Multisensor data fusion*. CRC press. 57

Hall, D.L. & McMullen, S.A. (1992). Mathematical techniques in multisensor data fusion, artech house. *Inc., Norwood, MA*. 57

Heberlein, T. (2002). Tactical operations and strategic intelligence: Sensor purpose and placement. *Net Squared Inc, Tech. Rep. TR-2002-04.02*. 27, 28, 61

Hodges, J.L. & Cam, L.L. (1960). The poisson approximation to the poisson binomial distribution. *The Annals of Mathematical Statistics*, **31**, 737–740. 35

Hohn, N. & Veitch, D. (2006). Inverting sampled traffic. In *IEEE/ACM Transactions on Networking*, 68–80. 78

Hoque, N., Bhuyan, M.H., Baishya, R., Bhattacharyya, D. & Kalita, J. (2013). Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*. 6

Hsu, C.C. & Huang, Y.P. (2008). Incremental clustering of mixed data based on distance hierarchy. *Expert Systems with Applications*, **35**, 1177–1185. 14

IETF (2009). Reducing redundancy in ip flow information export (ipfix) and packet sampling (psamp) reports. `http://ops.ietf.org/psamp/`. 77

Ishibashi, K., Kawahara, R., Tatsuya, M., Kondoh, T. & Asano, S. (2007). Effect of sampling rate and monitoring granularity on anomaly detectability. In *In 10th IEEE Global Internet Symposium 2007*. 78, 79, 88

Janakiram, D., Adi Mallikarjuna Reddy, V. & Phani Kumar, A. (2006). Outlier detection in wireless sensor networks using bayesian belief networks. In *Communication System Software and Middleware, 2006. Comsware 2006. First International Conference on*, 1–6, IEEE. 27

Jedwab, J. & Phaal, P. (1992). Traffic estimation for the largest sources on a network, using packet sampling with limited storage. Tech. rep., Jonathan Jedwab and Peter Phaal. 88

Jiang, G. & Cybenko, G. (2004). Temporal and spatial distributed event correlation for network security. In *American Control Conference, 2004. Proceedings of the 2004*, vol. 2, 996–1001, IEEE. 9, 10, 29

John, A. & Sivakumar, T. (2009). Ddos: Survey of traceback methods. *International Journal of Recent Trends in Engineering*, **1**, 241–245. 60

Joshi, M.V., Watson, I.T.J. & Agarwal, R.C. (2001). Mining needles in a haystack: Classifying rare classes via two-phase rule induction. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, **30**. 18

Jurga, R.E. & Hulb, M.M. (2007). Packet Sampling for Network Monitoring. Tech. rep., CERN HP Procurve openlab project, CH-1211, Gen'va 23, Switzerland. 78, 80, 88

Kandias, M., Mylonas, A., Virvilis, N., Theoharidou, M. & Gritzalis, D. (2010). An insider threat prediction model. In *Trust, Privacy and Security in Digital Business*, 26–37, Springer. 15, 16, 26

Kayacik, H.G., Zincir-Heywood, A.N. & Heywood, M.I. (2005). Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In *Proc. of the Third Annual Conference on Privacy, Security and Trust*. 20, 22, 39

KDD (1999). Kdd cup 1999 data. 20, 21, 22, 39

Khreich, W., Granger, E., Miri, A. & Sabourin, R. (2011). Adaptive ensembles of hmms applied to anomaly detection. *Pattern Recognition (Elsevier Science)*, **July 19, 2011**. 17

Kjaerulff, U. (1994). Reduction of computational complexity in bayesian networks through removal of weak dependences. In *Proceedings of the tenth international conference on uncertainty in artificial intelligence*, 374–382, Morgan Kaufmann Publishers Inc. 56

Krishnamurthy, S. & Sen, A. (2001). Stateful intrusion detection system (sids). In *Proceedings of the 2 nd International Information Warfare and Security Conference, Perth, Australia*. 25

Laskov, P., Gehl, C., Kruger, S. & Muller, K. (2006). Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, **7**, 19091936. 17

Lazarevic, A. (2013). Anomaly detection / outlier detection in security applications. http://www-users.cs.umn.edu/~aleks/anomaly_detection.htm. 33

Lazarevic, A., Ertoz, L., Kumar, V., Ozgur, A. & Srivastava, J. (2003). A comparative study of anomaly detection schemes in network intrusion detection. *Proc. SIAM*. 10

Le Cam, L. (1960). An approximation theorem for the poisson binomial distribution. *Pacific Journal of Mathematics*, **10**, 1181–1197. 35

Lee, W. & Stolfo, S.J. (1998). Data mining approaches for intrusion detection. In *Proc. of the T998 USENIX Security Symposium*. 21

Lippmann, R.P., Fried, D.J., Graf, I., Haines, J.W., Kendall, K.R., McClung, D., Weber, D., Webster, S.E., Wyschogrod, D., Cunningham, R.K. *et al.* (2000). Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, vol. 2, 12–26, IEEE. 20

Lu, N., Khoa, D. & Chawla, S. (2011). Online anomaly detection systems using incremental commute time. *CoRR*, **abs/1107.389**. 17

Mai, J., Chuah, C.N., Sridharan, A., Ye, T. & Zang, H. (2006a). Is sampled data sufficient for anomaly detection? In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, 165–176, ACM, New York, NY, USA. 78, 79

Mai, J., Sridharan, A., nee Chuah, C., Zang, H. & Ye, T. (2006b). Impact of packet sampling on portscan detection. *NATIONAL UNIVERSITY OF SINGAPORE, SINGAPORE IN*, **24**, 2285–2298. 78, 79

Marceau, C. (2000). Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the 2000 workshop on New security paradigms*, NSPW '00, 101–110, ACM, New York, NY, USA. 38

McHugh, J. (2000). The 1998 lincoln laboratory ids evaluation. In *Recent Advances in Intrusion Detection*, 145–161, Springer. 22, 42

McHugh, J. (2001). Intrusion and intrusion detection. *International Journal of Information Security*, 1415. 1

Mitropoulos, S., Patsos, D. & Douligeris, C. (2005). Network forensics: towards a classification of traceback mechanisms. In *Security and Privacy for Emerging Areas in Communication Networks, 2005. Workshop of the 1st International Conference on*, 9–16, IEEE. 60

Morrill, D. (2006). Cyber Conflict Attribution and the Law. http://it.toolbox.com/blogs/managing-infosec/cyber-conflict-attribution-and-the-law-10949. 60

Mullins, M. (2013). Defend your network from slow scanning. http://www.techrepublic.com/blog/security/defend-your-network-from-slow-scanning/361. 2, 11

Parker, T. (2010). Finger pointing for fun, profit and war? the importance of a technical attribution capability in an interconnected world. 60

Patcha, A. & Park, J.M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, **51**, 3448–3470. 1, 6, 12, 13, 33

Peng, T., Leckie, C. & Ramamohanarao, K. (2004). Proactively detecting distributed denial of service attacks using source ip address monitoring. In *NETWORKING 2004. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*, 771–782, Springer. 91

Phaal, P. (2013). Detecting nat devices using sflow. "http://www.sflow.org/detectNAT/. 88

ProQueSys (2012). Flowtraq, for effective monitoring, security, and forensics in a network environment. http://www.flowtraq.com/corporate/product/flowtraq. 27

R Development Core Team (2010). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0. 82

Rao, J. & Scott, A. (1981). The analysis of categorical data from complex sample surveys: chi-squared tests for goodness of fit and independence in two-way tables. *Journal of the American Statistical Association*, **76**, 221–230. 85

Rasoulifard, A., Bafghi, A.G. & Kahani, M. (2008). Incremental hybrid intrusion detection using ensemble of weak classifiers. *in Communications in Computer and Information Science*, **6**, 577584. 17

Reeves, J. & Panchen, S. (2002). Traffic monitoring with packet-based sampling for defense against security threats. *InMon Technology Whitepaper*. 78

Ren, F., Hu, L., Liang, H., Liu, X. & Ren, W. (2008). Using density-based incremental clustering for anomaly detection. In *Proc. of the 2008 International Conference on Computer Science and Software Engineering. Washington*. 14, 17

Riley, G.F. & Henderson, T.R. (2010). The ns-3 network simulator. In *Modeling and Tools for Network Simulation*, 15–34, Springer. 40

Saalbach, K. (2011). Cyberwar methods and practice. *Available FTP: dirk-koentopp. com Directory: download File: saalbach-cyberwar-methods-and-practice. pdf*. 60

Sager, G. (1998). Security fun with ocxmon and cflowd. *Presentation at the Internet*, **2**. 61

Satten, C. (2007). Lossless gigabit remote packet capture with linux. http://staff.washington.edu/corey/gulp/. 74

Sawilowsky, S.S. (2003). You think you've got trivials? *Journal of Modern Applied Statistical Methods*, **2**, 21. 84

Sebyala, A.A., Olukemi, T. & Sacks, L. (2002). Active platform security through intrusion detection using naive bayesian network for anomaly detection. In *London Communications Symposium*, Citeseer. 27

Shaikh, S.A., Chivers, H., Nobles, P., Clark, J.A. & Chen, H. (2009). Towards scalable intrusion detection. *Network Security*, **2009**, 12–16. 76

Sheth, C. & Thakker, R. (2011). Performance evaluation and comparative analysis of network firewalls. In *Devices and Communications (ICDeCom), 2011 International Conference on*, 1–5, IEEE. 9

Shiravi, A., Shiravi, H., Tavallaee, M. & Ghorbani, A.A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, **31**, 357–374. 21, 42

Siaterlis, C. & Maglaris, B. (2004). Towards multisensor data fusion for dos detection. In *Proceedings of the 2004 ACM symposium on Applied computing*, 439–446, ACM. 27, 57, 102

Smith, L.I. (2002). A tutorial on principal components analysis. *Cornell University, USA*, **51**, 52. 82

Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Schwartz, B., Kent, S.T. & Strayer, W.T. (2002). Single-packet ip traceback. *IEEE/ACM Transactions on Networking (ToN)*, **10**, 721–734. 61

Spafford, S.K.E.H. & Kumar, S. (1994). An application of pattern matching in intrusion detection. Tech. rep., Technical Report CSD-TR-94-013. Department Computer of Science, Purdue University. 13, 21, 23

StatSoft (2014). How To Identify Patterns in Time Series Data: Time Series Analysis. http://www.statsoft.com/Textbook/Time-Series-Analysis#systematic. 36, 37

Stefan, S., David, W., Anna, K. & Tom, A. (2001). Network support for ip traceback. *IEEE/ACM TRANSACTIONS ON NETWORKING*, **9**, 226–237. 61

Stone, R. *et al.* (2000). Centertrack: An ip overlay network for tracking dos floods. In *Proceedings of the 9th USENIX Security Symposium*, vol. 9, 199–212. 61

Streilein, W.W., Cunningham, R.K. & Webster, S.E. (2002). Improved detection of low profile probe and novel denial of service attacks. In *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*. 27

Tanase, M. (2002). One of these things is not like the others: The state of anomaly detection. http://www.symantec.com/connect/articles/one-these-things-not-others-state-anomaly-detection. 33

Tavallaee, M., Lu, W., Iqbal, S.A. & Ghorbani, A.A. (2008). A novel covariance matrix based approach for detecting network anomalies. In *Communication Networks and Services Research Conference, 2008. CNSR 2008. 6th Annual*, 75–81, IEEE. 5, 6, 17

Taylor, C. & Alves-Foss, J. (2001). Nate: N etwork analysis of a nomalous t raffic e vents, a low-cost approach. In *Proceedings of the 2001 workshop on New security paradigms*, 89–96, ACM. 78

Tellenbach, B., Brauckhoff, D. & May, M. (2008). Impact of traffic mix and packet sampling on anomaly visibility. In *Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, ICIMP '08, 31–36, IEEE Computer Society, Washington, DC, USA. 78, 79, 89

Theiler, J.P. & Cai, D.M. (2003). Resampling approach for anomaly detection in multispectral images. In *AeroSense 2003*, 230–240, International Society for Optics and Photonics. 18

Tozal, M.E. & Sarac, K. (2012). Estimating network layer subnet characteristics via statistical sampling. In *NETWORKING 2012*, 274–288, Springer. 74, 84

Trek, S. (n.d). Statistics and Probability Dictionary. http://stattrek.com/statistics/dictionary.aspx?definition=Optimum_allocation. 80, 81

Valdes, A. & Skinner, K. (2000). Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection*, 80–93, Springer. 27

Vallentin, M., Sommer, R., Lee, J., Leres, C., Paxson, V. & Tierney, B. (2007). The nids cluster: Scalable, stateful network intrusion detection on commodity hardware. In *Recent Advances in Intrusion Detection*, 107–126, Springer. 25

van Riel, J. & Irwin, B. (2006a). Toward visualised network intrusion detection. In *Proceedings of 9th Annual Southern African Telecommunication Networks and Applications Conference (SATNAC2006). Spier Wine Estate, Western Cape, South Africa*, 3–6. 18, 19

van Riel, J.P. & Irwin, B. (2006b). Identifying and investigating intrusive scanning patterns by visualizing network telescope traffic in a 3-d scatter-plot. In *ISSA*, 1–12. 9, 19, 84

Vasiliadis, G., Polychronakis, M. & Ioannidis, S. (2011). Midea: a multi-parallel intrusion detection architecture. In *Proceedings of the 18th ACM conference on Computer and communications security*, 297–308, ACM. 25

Vokorokos, L., Chovanec, M., Látka, O. & Kleinova, A. (2008). Security of distributed intrusion detection system based on multisensor fusion. In *Applied Machine Intelligence and Informatics, 2008. SAMI 2008. 6th International Symposium on*, 19–24, IEEE. 102

Wasilewska, A. (2010). Bayesian Networks. http://www.cs.sunysb.edu/ cse634/presentations/Bayes-datamining-presentation-06.ppt. 56

Wheeler, D.A. & Larsen, G.N. (2003). Techniques for cyber attack attribution. Tech. rep., DTIC Document. 60

Whyte, D., van Oorschot, P.C. & Kranakis, E. (2006). Exposure maps: removing reliance on attribution during scan detection. In *Proceedings of the 1st USENIX Workshop on Hot Topics in Security*, HOTSEC'06, USENIX Association, Berkeley, CA, USA. 91, 99, 102

Wool, A. (2006). Packet filtering and stateful firewalls. *Handbook of Information Security*, **3**, 526–536. 9

Yang, L. & Michailidis, G. (2007). Sampled based estimation of network traffic flow characteristics. In *SINFOCOM 2007*, 1775–1783. 78

Yankov, D., Keogh, E. & Rebbapragada, U. (2007). Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, 381–390, IEEE. 36

Yankov, D., Keogh, E. & Rebbapragada, U. (2008). Disk aware discord discovery: finding unusual time series in terabyte sized datasets. *Knowledge and Information Systems*, **17**, 241–262. 36

Ye, N., Xu, M. & Emran, S. (2000). Probabilistic networks with undirected links for anomaly detection. In *IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, 175–179. 27

Yegneswaran, V., Barford, P. & Ullrich, J. (2003). Internet intrusions: global characteristics and prevalence. In *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, 138–147, ACM. 1

Yi, Y., Wu, J. & Xu, W. (2011). Incremental svm based on reserved set for network intrusion detection. *Journal of Expert Systems with Applications*, **38**. 17

Yin, X., Yurcik, W., Treaster, M., Li, Y. & Lakkaraju, K. (2004). visflowconnect: netflow visualizations of link relationships for security situa-

tional awareness. In *Proc. of the 2004 ACM Workshop on visualization and Data Mining for Computer Security*, 35–44. 19

Yu, W.Y. & Lee, H. (2009). An incremental-learning method for supervised anomaly detection by cascading service classifier and iti decision tree methods. In *Proc. of the Pacific Asia Workshop on Intelligence and Security Informatics. Berlin*. 17

Zalewski, M. (2013). p0f v3 (version 3.06b). http://lcamtuf.coredump.cx/p0f3/. 88

Zhang, N.L. & Poole, D. (1994). A simple approach to bayesian network computations. http://repository.ust.hk/dspace/bitstream/1783.1/757/1/canai94.pdf. 56

Zhong, C. & Li, N. (2008a). Incremental clustering algorithm for intrusion detection using clonal selection. In *Computational Intelligence and Industrial Application, 2008. PACIIA'08. Pacific-Asia Workshop on*, vol. 1, 326–331, IEEE. 14

Zhong, C. & Li, N. (2008b). Incremental clustering algorithm for intrusion detection using clonal selection. In *Proc. of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application. Washington*. 14